TEKNILLINEN KORKEAKOULU

Tietotekniikan osasto

ROBOTIN ADAPTIIVINEN OHJAUS

Diplomityön aihe on hyväksytty Tietotekniikan osaston

osastoneuvoston kokouksessa 19.9.2005

Työn valvoja: TkT, prof. Juha Tuominen

Työn tekijä: Matti Pitkälä

Lahdessa toukokuun 30. päivänä 2006

Matti Pitkälä
Hinaajakatu 8 A 1
15140 Lahti

TEKNILLINEN KORKEAKOULU Tietotekniikan osasto	DIPLOMITYÖN TIIVISTELMÄ
Tekijä Matti Pitkälä	Päiväys 30.5.2006
	Sivumäärä 123
Työn nimi ROBOTIN ADAPTIIVINEN OHJAUS	
Professuuri Teollisuuden tietotekniikka	Koodi 2173
Työn valvoja Prof. Juha Tuominen	
Työn ohjaaja DI Timo Turunen	

Ulkoisten anturitietojen käytöllä voidaan lisätä robotin adaptiivisuutta, mikä mahdollistaisi uusia käyttösovelluksia robotisoinnin yhteydessä. Vaativin tavoite adaptiivisessa ohjauksessa on luoda reaaliajassa uutta robotin liikerataa. Tämän tutkimustyön tarkoituksena oli selvittää tämän kaltaisten tutkimustöiden taustoja ja myös itse käytännössä toteuttaa, kuinka ulkoisilla antureilla voitaisiin parantaa ABB IRB 4400 -robotin ohjauksen adaptiivisuutta.

Tutkimustyön tärkeimpänä yksittäisenä tavoitteena oli selvittää ulkoisten voima-antureiden käyttöä robotin ohjauksessa, jolloin teollisuusrobotilla olisi myös jonkin asteinen ulkoinen tuntoaisti. Tämän kaltaisilla sovelluksilla olisi käyttöä esimerkiksi puuteollisuudessa robotilla suoritettavissa hiontatehtävissä, joissa on ollut ongelmana sekä robotin liikeradan luominen että hiontajälki.

Etenkin Ruotsissa on suoritettu vastaavia tutkimuksia puuteollisuuden sovelluksiin ja niissä kaikissa on ollut suurin este onnistuneen lopputuloksen saamiseksi robottiohjaimen puutteellinen laskentateho reaaliaikaisen radan muodostamisessa. Niin tapahtui myös tässä tutkimustyössä, jonka tuloksissa selvisi se, että ABB:n S4C – robottiohjain kykenee muodostamaan uutta liikerataa ainoastaan 0,2...0,3 sekunnin välein, mikä on aivan liian hidas toiminto.

Tutkimustyön toissijaisena tavoitteena oli tutkia sitä, että voidaanko tätä robottiohjaimen vajavaista kykyä muodostaa reaaliaikaista rataa jotenkin parantaa. Robotin yhteyteen lisättiin uuden sukupolven ohjelmoitava logiikka varustettuna sumean säädön yksiköllä ja tämä ei aiheuttanut minkäänlaisia ongelmia nopeuden osalta anturitietojen käsittelyssä. Anturitietojen käsittely sumealla säädöllä helpotti oleellisesti käytännön toimia koeajojen suorittamisessa, mutta itse radan muodostamisen lopputulokseen sillä oli kuitenkin hyvin vähäinen merkitys. Robotin liikkeen värähtely saatiin aavistuksen verran vähäisemmäksi. Myös erilaisten PC – ohjelmistojen käyttöä robotin ohjauksen yhteydessä toteutettiin tutkimustyön käytännön järjestelyissä. Reaaliaikaisen robottiradan muodostaminen on kuitenkin tulevaisuudessa ja osittain jo nyt mahdollista, koska tämän päivän robottiohjaimien laskentakyky on aivan eri tasolla kuin tutkimustyössä käytetty 90-luvun loppupuolen ohjaimen. Tulevaisuus näyttää siis erittäin lupaavalta robotin adaptiivisen ohjauksen lisääntyvälle käytölle.

Avainsanat
robotin adaptiivinen ohjaus, sumea säätö, voima-anturiohjaus

HELSINKI UNIVERSITY OF TECHNOLOGY Department of Computer Science and Engineering	ABSTRACT OF MASTER'S THESIS

Author	Date
Matti Pitkälä	30 May 2006
	Pages 123

Title of thesis

ADAPTIVE ROBOT CONTROL

Professorship	Professorship Code
Information Technology in Industrial Production	2173

Supervisor

Prof. Juha Tuominen

Instructor

M.Sc. Timo Turunen

By using external sensors it is possible to increase adaptivity in robot control, which will make it possible to use robots in new applications. The most complex question in adaptive robot control is what kind of ability for real-time control a robot has. The objective of this work was to study the background of real-time control and to develop a control system for an ABB IRB 4400 robot, based on using external sensors to improve its adaptive control.

The most important separate objective of this work was to study possibilities for using external force sensors. This kind of application will make it possible for an industrial robot to have a sense of touch. This would be particularly useful introducing robots for sanding applications in the woodworking and furniture industries.

Especially in Sweden, a lot of research has been done on applications in the woodworking and furniture industries and the results have shown that the robot control systems do not have enough capacity to calculate robot movements in real time.

The robot control unit used in the test cases of this thesis was able to calculate a new robot target only at 0.2…0.3 s intervals to create the robot path in real time. This is not enough to create a smooth robot path.

The secondary objective of this work was to examine if there are any methods to improve the process of creating the real-time robot path. The system produced in this work consisted of a new generation programmable logic controller including a fuzzy-logic control unit. No problems arose concerning the sensor refresh time between the PLC and the robot control unit. The system makes it easier to carry out practical tests, but it does not make the robot path as smooth as desired. The vibration of movement was only a little smaller.

In the future and even nowadays it is possible to get better results in adaptive robot control, because the calculation ability of the computers used in robots is much better in new generation robot control units than before.

Keywords

adaptive robot control, fuzzy-logic control, sensor, force feedback, real time

ALKUSANAT

Diplomityö on tehty Teknillisen Korkeakoulun tietotekniikan osastolla ja Lahden ammattikorkeakoulun Tekniikan laitoksen robottilaboratorion toimeksiannosta.
Työn tarkastajana on toiminut Teknillisen Korkeakoulun puolesta TkT prof. Juha Tuominen ja ohjaajana DI Timo Turunen. Kiitän diplomityön tarkastajaa ja valvojaa saamistani neuvoista.
Lisäksi haluan kiittää Lahden ammattikorkeakoulun Tekniikan laitoksen laboratorio-mekaanikkoa Jari Kukkosta suoritetuista koneistuksista valmistettaessa suunnittelemaani voima-anturisovitinta tutkimustyössä käytetyn robotin ranteen laippaan. Suomen ABB:n robottiosaston Ville Raulaa, Teemu Rantalaa ja Oskari Hakaluotoa haluan myös erityisesti kiittää sekä omasta että työn toimeksiantajan puolesta myötämielisestä suhtautumisesta projektin vaatimissa taloudellisissa hankinnoissa ja tutkimustyön yhteydessä tarvituista erikoismanuaaleiden toimittamisesta tutkimustyön tekijälle.

Lahdessa 30.5.2006
Matti Pitkälä

SYMBOLILUETTELO JA KÄYTETYT LYHENTEET

ABB	Allmänna Svenska Elektriska Aktiebolaget Brown Boveri
AD-muunnin	Analoginen tuloyksikkö, analoginen viesti digitaaliseksi viestiksi
BCD	Binary Coded Decimal
CNC	Computer Numerically Controlled
CPU	Central Processor Unit
DOF	Degrees Of Freedom (Vapausasteet)
DSP	Digital Signal Processing (Digitaalinen signaalin käsittely)
FTP	File Transfer Protocol
Fuzzy Control	Sumea säätö
ISO	International Standardization Organization
I/O	Input / Output (Sisääntulot / ulostulot)
IRB	Industrial Robot
NFS	Network File Server
PCI	Periferal Component Interconnect
Pitch	Kierto robotin ranteen y – akselin ympäri
PLC	Programmable Logic Controller (Ohjelmoitava logiikka)
PUMA	Programmable Universal Machine Arm
RAM	Random Access Memory
RAP	Robot Application Protocol
Roll	Kierto robotin ranteen z – akselin ympäri
rpm	revolution per minutes (kierroksia minuutissa)
RS232	Recommended Standard 232
S4C	ABB teollisuusrobottiohjain
SCARA	Selective Compliance Assembly Robot Arm
SDK	Software Development Kit
TKP	Robotin työkalupiste
Yaw	Kierto robotin ranteen x – akselin ympäri

SISÄLLYSLUETTELO

1. JOHDANTO

1.1 TAUSTAA

Teollisuusrobotit ovat vähitellen alkaneet lisääntyä puuteollisuudessa tosin hyvin verkkaisesti. Lahden ammattikorkeakoulun Tekniikan laitoksella (entinen Lahden teknillinen oppilaitos) on tutkittu vuodesta 1991 lähtien teollisuusrobottien mahdollisia käyttökohteita huonekaluteollisuudessa. Tekes eli Teknologian kehittämiskeskus oli 1990–luvun alussa tukemassa useita robotisoinnin tutkimushankkeita, jotka olivat osaprojekteja huonekaluteollisuuden kehittämishankkeissa (HUKE). Näissä projekteissa tutkittiin robotin soveltuvuutta puutuotteiden kokoonpanoon ja myös robotin käytön mahdollisuuksia numeerisesti ohjatun eli CNC–jyrsinkoneen yhteydessä. Osa näistä projekteista johti myös onnistuneeseen lopputulokseen, mikä tarkoitti myös sitä, että vastaavanlainen sovellus otettiin teollisuudessa käyttöön.
1990–luvun lopussa suurin mielenkiinto robotin käyttömahdollisuuksien tutkimisessa puuteollisuudessa oli puutuotteiden hionta, mutta nämä projektit eivät kuitenkaan olleet kovinkaan onnistuneita, koska vaadittava hiontajälki ei vastannut odotuksia. Suuri mielenkiinto hionnan robotisointiin johtuu siitä, että käsin suoritettu hionta työvaiheena on työläs ja yksitoikkoinen, mikä aiheuttaa työvoiman saatavuuden kannalta ongelmia juuri hiontatehtäviin. Tämän kaltaiset tehtävät vaativat tuntoaistin lisäämistä robottiin, mikä tarkoittaa käytännössä voimaan perustuvan ohjauksen käyttöä. Voima-antureiden käyttöä robotin ohjauksessa on tutkittu usean vuoden ajan ja niissä kaikissa on tullut ilmi se tosiasia, että robottiohjaimien tehokkuus on ollut riittämätön reaaliaikaisen radan muodostamiseen. Mutta viimeaikainen kehitys robottiohjaimissa on parantanut huomattavasti tätä ominaisuutta ja on todennäköistä lähitulevaisuudessa, että voimaan perustuva ohjaus yleistyy myös käytännön sovelluksissa, eikä pelkästään tutkimustöissä.
Toinen ongelma robotilla toteutettavissa puutuotteiden hiontatehtävissä on tarvittavan liikeradan luominen, koska puutuotteissa on paljon mittapoikkeamia eli tarkkuus tuotteiden valmistuksessa ei ole lähellekään samaa luokkaa kuin esimerkiksi metalliteollisuuden tuotteilla on. Tätä voidaan parantaa normaaleilla antureilla ja näin lisätä niiden avulla ohjauksen adaptiivisuutta.
Tämän diplomityön tarkoitus on adaptiivisuuden lisäämisen ja voimaan perustuvan ohjauksen lisäksi tutkia sitä, että voidaanko robotin suorituskykyä juuri radan uudelleen muodostamisessa parantaa sumean säädön (Fuzzy–logiikka) avulla.

Puuteollisuuden robotisoinnin suurimmat mielenkiinnonkohteet ovat olleet jyrsintä, hionta, puun leikkaus ja kokoonpano. Teollisuusrobottien käyttöä jyrsinnässä voidaan katsoa täydentävänä keinona CNC–koneiden käytölle, kun tarvitaan suurta joustavuutta sovelluksissa. Esimerkiksi robottisoluissa joihin halutaan jyrsinnän lisäksi porausta, hiontaa, liimausta ja kokoonpanoa, robottien käytöllä voidaan lisätä huomattavasti tehokkuutta. Tuotteiden mittatoleranssivaatimukset ovat huomattavasti alhaisemmat verrattuna muihin mekaanisen teollisuuden tuotteissa oleviin. Harvoin tai oikeastaan koskaan puutuotteissa ei vaadita parempaa tarkkuutta, johon normaalit teollisuusrobotit eivät kykene eli 0.1 mm:n tarkkuutta. Mutta miksi sitten ollaan kiinnostuneita ohjauksesta, joissa hyödynnetään antureita. Eräs negatiivisista seikoista, joita halutaan välttää, on robotin dynaaminen käyttäytyminen jyrsintäparametrien muuttuessa. Antureilta tulevan tiedon avulla voidaan kompensoida myös paras mahdollinen leikkuunopeus verrattuna haluttuun pinnan sileyteen. Tämä on erityisen tärkeätä juuri

silloin, kun puun syiden suunta vaihtelee tai / ja kohdataan oksakohtia puussa. Adaptiivisuuden käytölle tällaisissa sovelluksissa on kuitenkin kaksi vaikeata asiaa ratkaistavaksi. Toinen on leikkuuparametrien eli lähinnä nyt leikkuuvoiman näytteen ottopäivitystaajuus, minkä suorittaminen on erittäin haastava tehtävä. Esimerkiksi jyrsimen pyörimisnopeuden ollessa 10000 rpm ja vastaavasti 30 asteen kaarikulmassa syntyvän lastun pituuden mittaamiseen tarvitaan 8000 Hz:n näyttöön ottotaajuutta, jotta saataisiin neljä arvoa jokaisella kosketuskerralla. Toiseksi jo esillä ollut robotin kyky päivittää omaa liikerataansa on rajallinen. Esimerkiksi oksan havaitsemiseen 1 mm etäisyydellä 2 m/min liikenopeudella vaaditaan 33 Hz:n päivitystaajuus, johon perinteiset robottiohjaimet eivät ole kyenneet. [9]

Puuteollisuuden tuotteiden valmistuksessa on paineita kyetä vastamaan laajempaan tuotetarjontaan ja lyhyemmillä tilausajoilla. Puutuotteiden valmistus on usein automatisoinnissa vaativa kohde, koska suhteellinen kosteus ja puulaji vaikuttavat valmistusprosessiin. Teollisuusrobottien käyttöönotto puutuotteiden valmistuksessa voidaan nähdä keinona lisätä automaatiota ja samanaikaisesti säilyttää tuotannon joustavuus myös pienillä valmistussarjoilla.
Teollisuusrobotit ovat siis vähitellen alkaneet lisääntyä puuteollisuudessa tosin hyvin verkkaisesti. Sovellukset perustuvat yleensä tarkkaan paikoitukseen, jollaisia ovat paletointi, pakkaus sekä maalaustehtävät ja huonekalujen metalliosien hitsaus-sovellukset. Ominaista näille sovelluksille on myös se, että ne vaativat suuria tuotteiden valmistusmääriä, jolloin ne ovat myös suhteellisen harvinaisia, koska vain pieni osa tuotantolaitoksista kykenee niihin. Erityyppiset sovellukset on tehty tuoteosien käsittelyyn kuten CNC-koneiden panostukseen ja purkamiseen, tuotteiden pakkaamiseen ja paletointiin jigeihin sekä myös yksinkertaisiin hiontatehtäviin. Adaptiivisuudella tässä yhteydessä tarkoitetaan mahdollisuutta kompensoida antureilla puutuotteiden mittavaihteluja ja ympäristöstä aiheutuvia muutoksia. Adaptiivisuus kytketään robotin paikoitukseen ja sen liikenopeuteen. Tämän tutkimustyön tarkoituksena on myös selvittää vaatimukset käytettäville antureille.

1.2 YLEISIÄ NÄKÖKOHTIA TUTKIMUSKOHTEESTA

Samoin kuin ihminen käyttää aistejaan jatkuvasti ja samanaikaisesti, myös robotille on pyrittävä saamaan samankaltaisia toimintoja useiden ulkoisten antureiden avulla, jolloin sen toimintaan saadaan huomattavasti joustavuutta lisää. Vaikeutena ja rajoittavana tekijänä on ollut suhteellisen alhainen näyttöön ottotaajuus, johon robottien ohjaimet ovat kyenneet. Myös robotin radan uudelleen määrittelyn hitaus aiheuttaa rajoitukset sisäisen aikaviiveen vuoksi. ABB IRB 2400/10 -robotin käyttäessä RS232–kommunikointia saadaan ainoastaan 5 Hz:n taajuus robotin radan päivitykselle ja vaikka käytettäisiin nopeampiakin väylätekniikoita on maksimi taajuus 10 Hz. [8]
Kappaleiden tunnistamiseen ja paikantamiseen optiset anturit soveltuvat erinomaisesti. Kappaleiden laadun tunnistamisessa ja muissa vaativimmissa sovelluksissa voidaan käyttää CCD-kameroita, spektrometreja, lasereita ja ultraääniantureita.
Voima- ja momenttiantureiden mahdollisella käyttöönotolla nähdään myös suuria odotuksia teollisuusrobottien kehityksessä ja soveltamisessa puuteollisuudessa. Mahdollisia sovellusalueita, joissa voitaisiin hyödyntää voimaohjausta, olisivat puutuotteiden kokoonpano-, hionta- ja kiillotustehtävissä tapahtuva robotin paikoituksen määrittely. Ongelmana on kuitenkin ollut robotin liikenopeus, sillä maksimissaan 50 mm/s:n nopeutta on voitu käyttää tämän kaltaisissa tehtävissä. Tähän

on syynä ollut robotin ohjaimen vajavainen kyky päivittää uutta käsivarren rataa. Lähitulevaisuudessa ja jo nyt on olemassa oleellisesti nopeampia robottiohjaimia.
Puutuotteiden pinnan tasaisuus ja laatu lakkauksen, värjäyksen tai kiillotuksen jälkeen on suuresti riippuvainen edeltävästä työvaiheesta eli normaalisti hionnasta. Vaihtelevat tekijät kuten aika, nauhan nopeus, hiomajyvän koko ja terävyys, hiontapaine sekä hiomamateriaali määräävät poistettavan ainemäärän hiontaprosessissa.
Vaikeudet kontrolloida näitä tekijöitä ovat olleet yleisesti syynä hiontaprosessien vaatimattomaan automaatioasteeseen. Hionta on ollut tärkeä ja perinteisesti paljon työvoimaa vaativa työvaihe puutuotteiden valmistuksessa. Käytetyissä robotti-hiontasovelluksissa puuteollisuudessa on luotettu joustavaan työstöpäähän kompensoimaan puussa olevia vaihteluita, jotka vaikuttavat hiomiseen. Robotin adaptiivisuuden lisäämisellä voitaisiin myös kompensoida juuri näitä poikkeamia. Tähän tehtävään juuri voima-antureiden käyttö antaisi aivan uuden lähestymismallin. Tähän asti voima-antureita on käytetty lähinnä passiivisesti eli rajaamassa voima vain tietylle välille eikä niinkään ole ollut mahdollista niiden avulla päivittää robotin liikerataa eli käyttää voima-ohjausta aktiivisesti.

Yksi aihe puutuotteiden valmistuksen automatisoinnissa on ollut puun leikkaus ja talttaus, koska on yhä vaikeampaa löytää riittävän taitavaa työvoimaa näiden tehtävien suorittamiseen. Koristeleikkauksen automatisointia on vaikeuttanut se, että perinteisen pyörivän jyrsimen käyttö ei ole mahdollista, koska sen avulla ei voida tehdä teräviä kulmia eikä reunoja lopputuotteeseen ja juuri nämä ovat oleellisia ominaisuuksia puhuttaessa käsin tehdystä korkealuokkaisesta lopputuotteesta. Näissä sovelluksissa voitaisiin yhdistää normaali jyrsintätyöstömenetelmä ja robotin käyttö puun leikkauksessa tehtäessä terävät reunat tuotteeseen. [15]
Puutuotteiden automaattisessa kokoonpanossa on ollut vaikeutena osien vaatimaton mittatoleranssiaste ja parantamalla ensin tätä seikkaa, voidaan käyttöön ottaa suhteellisen helposti robotilla tehtävää kokoonpanoa. Huonoa mittatoleranssia voidaan nyt myös kompensoida käytettäessä robotin yhteydessä voimaohjausta.

Tässä tutkimustyössä on paneuduttu myös käyttöliittymän luomiseen robotin ohjauksessa, koska robottien käytön lisäämiselle puuteollisuudessa vaaditaan myös helpompaa käyttöliittymää ja robotin ohjelmointia. Yleisesti voidaan sanoa, että normaalisti henkilökunnan tieto- ja taitotaso on puuteollisuudessa vaatimattomampaa kuin useilla muilla teollisuuden aloilla. Tässä on myös muistettava se, että vastaavasti taas näiden sovellusten kustannukset eivät saa nousta liian suuriksi näitä kehitettäessä. Käyttöliittymien kehitykseen on robottivalmistajilla omia kehitystyökaluja, joilla voidaan robottien perinteistä ohjelmointia jakaa eri hierarkkisille tasoille, mikä on ensiarvoisen tärkeätä juuri anturipohjaisissa ohjauksissa, koska nämä sovellukset vaativat suurta ohjelmointitaitoa, mikä taas vaikeuttaisi niiden yleistymistä käytännössä. Nyt voidaan kehittää kuitenkin graafisia käyttöliittymiä loppukäyttäjille ja piilottaa alemmalle tasolle vaativimmat kohdat näissä sovelluksissa.

1.3 TUTKIMUKSEN TAVOITTEET, RAKENNE JA TOTEUTUS

Tämän tutkimustyön tavoitteena oli selvittää, mitä mahdollisuuksia on olemassa kehittää teollisuusrobotin soveltuvuutta puusepänteollisuuden vaativimpiin robotisointikohteisiin kuten puutuotteiden hiontaan. Ensimmäinen tavoite oli löytää erilaisia mahdollisia keinoja lähteä parantamaan robotin soveltuvuutta ja joustavuutta. Tavoitteena ei siis ollut tutkia esimerkiksi puutuotteiden hiontaa tietyllä menetelmällä ja kuinka hyviä mahdollisia tuloksia sillä voitaisiin saavuttaa, vaan ennen kaikkea tutkia syvällisemmin itse robottia ja mahdollisen järjestelmän vaatimia muita komponentteja kuten antureita, ohjelmistoja ja muita ohjainlaitteita.
Alussa luodaan lyhyt katsaus erilaisiin robottirakenteisiin, joista voitaisiin valita sopivimmat robottirakenteet puuteollisuuden tarpeisiin. Tämä katsaus rakenteisiin on tehty, vaikka tutkimustyöntekijällä on jo yli kymmenen vuoden kokemus siitä, että kiertyvänivelinen robotti on oikea robottityyppi tutkimuksen kohteena olevalle puuteollisuuden robotisoinnille. Robotiikan matematiikkaa paikoitusten, kiertojen ja siirtojen toteuttamisen osalta haluttiin myös syvällisemmin selvittää, koska nykyiset robottiohjaimet omaavat myös vastaavia ohjelmointikäskyjä, jolloin on mahdollista myös käyttää tämän pohjalta syntyviä kehitysideoita.
Robotin tarrain ja siihen liitetyt anturit ovat erittäin oleellinen seikka robottisovellutuksen kannalta, minkä vuoksi on selvitetty mahdollisia antureita, jotka soveltuvat robotin adaptiiviseen ohjaukseen.
Nykyiset robottivalmistajat suosivat suljettua ohjausjärjestelmää, jolloin robotin sisäiseen toimintaan eli käyttöjärjestelmätasolle ei ole mahdollisuutta mennä tekemään muutoksia. Ainoastaan robotin systeemiparametriarvoja voidaan muuttaa ja näin myös tehtiin, jolloin robotin ominaisuuksia saatiin vähän parannettua kohteena olevalle käytölle. Ulkoisten antureiden mahdollista käyttöä robotin adaptiivisuuden lisäämiselle tutkittiin ennen kaikkea voima-antureiden ja laserantureiden osalta.
Lisäksi haluttiin selvittää saadaanko oleellista hyötyä liittämällä robotin yhteyteen nykyaikainen ohjelmoitava logiikka, jossa käytetään myös sumean säädön yksikköä. Tutkimuksen tekijällä on myös näiden laitteiden osalta yli 15 vuoden kokemus, mutta siitä huolimatta lyhyt katsaus ohjelmoitavien logiikoiden ja sumean säädön osalta on myös tehty.
Käytännön järjestelyt onnistuivat erinomaisesti laitteiden osalta Lahden ammattikorkeakoulun Tekniikan laitoksen robottilaboratoriossa ja myös tarvittavia hankintoja voitiin suorittaa laboratoriomäärärahojen puitteissa. Ajallisesti käytännön toteutukset veivät tutkimustyöhön käytetystä ajasta noin puolet ja kirjallinen osuus toisen puolen. Viitemateriaalin hankinta vastaavista tutkimuksista aloitettiin jo puoli vuotta ennen diplomityön virallista alkamisajankohtaa tieteellisen tiedon hakukurssin yhteydessä.
Yhteenvetona tutkimustyön toteutuksesta ja tavoitteesta voidaan todeta, että tavoitteena oli löytää parantavia ratkaisuja robotin käytölle puuteollisuudessa neljällä osa-alueella. Nämä neljä aluetta olivat 1) robotin tutkiminen, 2) antureiden ja sitä kautta adaptiivisuuden käyttö, 3) ohjelmoitavan logiikan ja sumean säädön käyttö ja 4) PC:n liittäminen robotin yhteyteen eli erilaisten ohjelmistojen hyödyntäminen. Nämä kaikki osa-alueet myös toteutettiin käytännössä. Tässä kirjallisessa osuudessa ne ovat viimeisessä luvussa myös analysoitu.

2. ROBOTIT

2.1 ROBOTTITYYPIT JA RAKENTEET

2.1.1 Yleistä

Teollisuusrobotteja on tähän mennessä valmistanut ainakin viisisataa yritystä. Kunkin valikoimaan on koko ajan kuulunut useita robottimalleja. Yhden mallin elinkaari on kestänyt keskimäärin neljä vuotta. Lisäksi rakenteita on jouduttu erilaistamaan patenttien ja eri sovellusten vuoksi. Joten erilaisia teollisuusrobotteja on suunniteltu useita tuhansia. Markkinoiden keskittyessä on vaihtoehtojen kirjo hieman supistunut, mutta jatkuvasti ilmaantuu uusiakin robottien valmistajia. [1]
Standardi ISO 8373 määrittelee teollisuusrobottien sanastoa ja myös yleisimmät robottimallit mekaanisen rakenteen mukaan. Merkittäviä robottivalmistajia ovat mm. ABB (Asea Brown Boveri), Motoman, Fanuc, KUKA, Hitachi ja Kawasaki.

Yleisimmät rakenteet ovat:

- suorakulmainen robotti
- sylinterirobotti
- napakoordinaatisto robotti
- scara-robotti
- kiertyvänivelinen robotti
- rinnakkaisrakenteinen robotti.

Robottien jakaminen voidaan myös tehdä tehtävän mukaan:

- maalausrobotit (nopeita, tarkkuus ei ole tärkeää)
- prosessirobotit (nopeita, tarkkoja, mutta "jäykkiä")
- kokoonpanorobotit (nopeita, suuri tarkkuus, alhainen kappaleenkäsittelykyky).

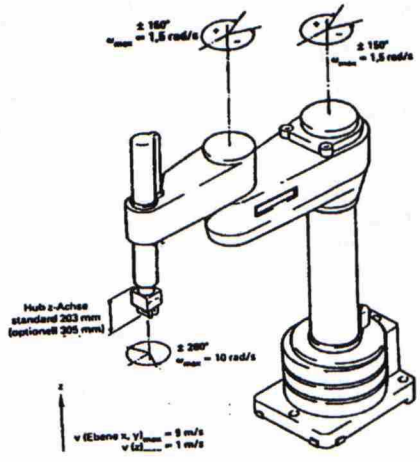
2.1.2 Robottityypit ja rakenteet

Suorakulmaiset robotit

Suorakulmaisten robottien kolme ensimmäistä vapausastetta ovat lineaarisia. Tyypillisintä edustajaa kutsutaan yleensä portaalirobotiksi. Sen rakenne on tuettu työalueen nurkista palkeilla.

Kuva 2.1 Yleiskuva portaalirobotista (ABB:n robottiesite).

Scara - robotit



Kuva 2.2 Scara – robotti (Kuivanen, R., Robotiikka kirja [1]).

Scara-robotissa (Selective Compliance Assembly Robot Arm) on tiettyyn suuntaan joustava kokoonpanorobottikäsivarsi ja kolmella kiertyvällä nivelellä työkalu saadaan tietyllä tasolla oikeaan kohtaan ja kiertymäkulmaan. Neljäs lineaarinen pystyliike on työtason normaalin suuntainen. Scara-robotti muistuttaa ihmisen vaakatasossa liikkuvaa käsivartta, mutta ranteeseen on asennettu pystyjohde.

Kiertyväniveliset robotit

Kiertyvänivelisessä robotissa kaikki vapausasteet ovat kiertyviä. Ne ovat tavallisimpia teollisuusrobotteja. Vapausasteita on yleensä kuusi tai neljä.

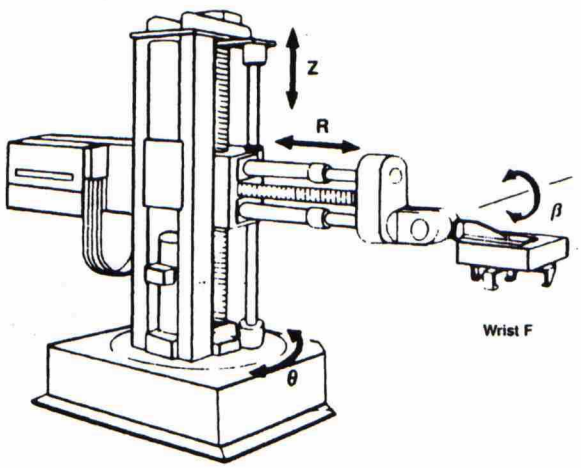
Kuva 2.3 ABB:n IRB 6400R kiertyvänivelinen teollisuusrobotti 100 kg:n kantokyvyllä (ABB:n robottiesite).

Nykyiset teollisuusrobotit perustuvat lähes poikkeuksetta tähän mekaniikkaan, jossa tukivarret on kytketty peräkkäin. Tästä johtuu, että robottien kuormankantokyky on melko pieni, mutta työalue eli ulottuvuus on suurehko. Kehitteillä ovat myös sellaiset

robottikäsivarret, joissa on yli kuusi vapausastetta paremman kurottelukyvyn saavuttamiseksi. Tällöin robotin on esimerkiksi mahdollista kurottua ikkuna-aukosta auton sisälle, liikuttaa työkalua halutulla tavalla ja samalla väistellä ikkunoiden reunoja.

Sylinterirobotti

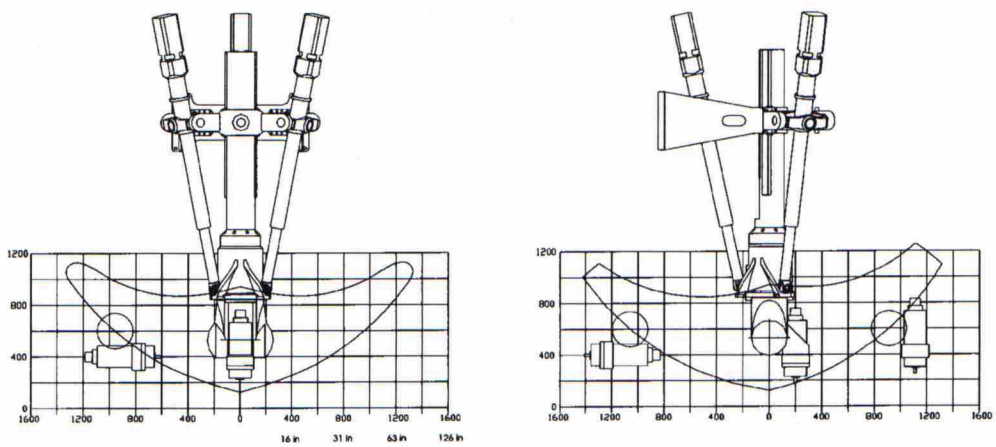
Sylinterirobotin nimitys on luonnollisesti peräisin sylinterikoordinaatistosta.



Kuva 2.4 Perinteinen sylinterirobottirakenne (Kuivanen, R. Robotiikka kirja [1]).

Rinnakkaisrakenteiset robotit

Suuria voimia robotit saadaan kestämään kytkemällä joitain robotin vapausasteita (tai pikemminkin toimilaitteita) rinnakkain. Tällöin rakennekin tukevoituu, mutta työalue rajoittuu pieneksi.



Kuva 2.5 Rinnakkaisrakenteinen robotti työstötehtäviin (Neos robottiesite).

Erästä perusratkaisua, jossa kahden levyn välistä asemaa muutetaan kuudella kumpaankin levyyn kytketyllä lineaarisella toimilaitteella, kutsutaan nimellä "Stewartin alusta". Erittäin nopeita robotteja valmistetaan kytkemällä rinnakkain ultrakevyitä rakenteita. Tällöin on mahdollista robotille saada aikaan erittäin nopeita liikkeitä, kunhan vain työkohteen ympärillä on riittävästi vapaata työaluetta.

Kuva 2.6 ABB:n IRB 340 robotti soveltuu hyvin elintarviketeollisuuden pakkauslinjalle (ABB:n robottiesite).

Suljetun kinemaattisen rakenteen idea on jakaa tukivoimat toisiaan tukevien rakenteiden avulla, jolloin robotista tulee kestävä. Keveys ja mahdollisuus suuriin voimiin ovatkin rakenteen suurimmat edut. Näitä robottirakenteita on tutkittu ja suunniteltu vasta 1990-luvulta alkaen. Ne ovat yleistymässä työstötehtävissä kuten karaa liikuttavina rakenteina.

Kuva 2.7 Työstöön tarkoitettu suljetun kinemaattisen rakenteen mukainen robotti (Tricept – robotti, Neos robottiesite).

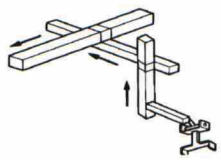
2.1.3 Robottien koordinaatistojärjestelmät ja kehykset

Robottien akselit (vapausasteet) eli nivelet voivat olla rakenteeltaan hyvin monenlaisia kuten suoraviivaisia, kiertyviä, pallomaisia tai liukuvia liikkeitä suorittavia. Kiertyvät ja suoraviivaiset liikkeet ovat robottien yleisimmät liikemuodot. Toimilaitteina lineaariliikkeille käytetään yleensä pneumaattisia tai hydraulisia sylintereitä sekä sähköisiä lineaaritoimilaitteita. Kiertyvät liikkeet on yleensä toteutettu sähköisillä servomoottoreilla, mutta myös sähköisillä askelmoottoreilla sekä pneumaattisilla tai hydraulisilla vääntömoottoreilla voidaan kiertoliikkeet toteuttaa.

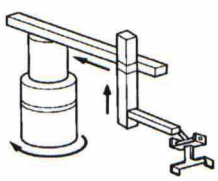
Robottien konfiguraatiossa sen käyttämä koordinaatistojärjestelmä on yhdenmukainen sen mekaanisen rakenteen kanssa.

Robottien yleisimmät koordinaatistojärjestelmät:

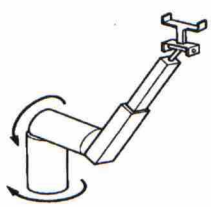
- Suorakulmainen (Cartesian), sisältää kolme lineaarista nivelakselia.



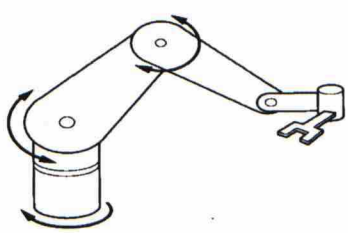
- Sylinterimäinen (Cylindrical), sisältää yhden kiertyvän ja kaksi lineaarista nivelakselia.



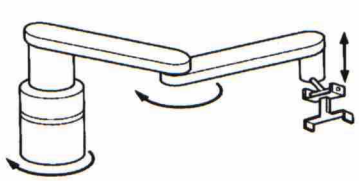
- Pallomainen (Spherical), sisältää kaksi kiertyvää ja yhden lineaarisen nivelakselin.



- Nivelmäinen (Articulated), sisältää kolme ihmiskäden kaltaista kiertyvää nivelakselia.



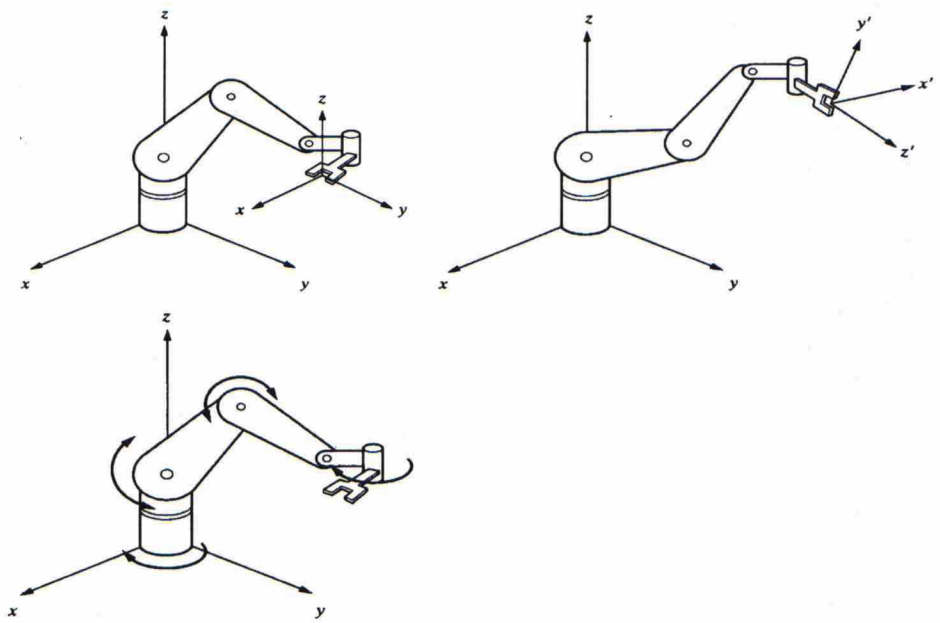
- SCARA (Selective Compliance Assembly Robot Arm), sisältää kaksi rinnakkaista vaakatason suuntaisen liikkeen mahdollistavaa kiertyvää nivelakselia sekä yhden pystysuoran suuntaisen liikkeen mahdollistavan lineaarisen nivelakselin.



SCARA

Kuva 2.8 Robottien yleisimmät koordinaatistojärjestelmät.[4]

Robotteja liikutetaan suhteessa erilaisiin koordinaatistokehyksiin. Liikkeiden toteutukset ovat erilaisia, riippuen minkälaisen koordinaatistokehyksen mukaan liike toteutetaan (kuva 2.8).
Maailmakoordinaatiston mukaisessa kohdekehyksessä robotti liikkuu pääakselien x-, y-, tai z -akselien suuntaisesti, jolloin samanaikaisesti voi useampi robotin nivelistä suorittaa oman liikkeensä. Nivel- eli peruskoordinaatiston mukaisessa kohdekehyksessä liikutetaan robotin jokaista yksittäistä niveltä erikseen. Työkalukoordinaatiston mukaisessa kohdekehyksessä on luotu työkalun mukainen koordinaatisto, jonka mukaisesti robotti suorittaa liikkeensä, mutta erona maailmankoordinaatiston mukaiseen liikuttamiseen on se, että työkalukoordinaatisto liikkuu myös robotin liikkeen mukana. Työkalukohdekehyksen käyttö on hyödyllistä robotin ohjelmoinnissa, jossa robotin on liikuttava eri kohteiden välillä tai osien kokoonpanossa.



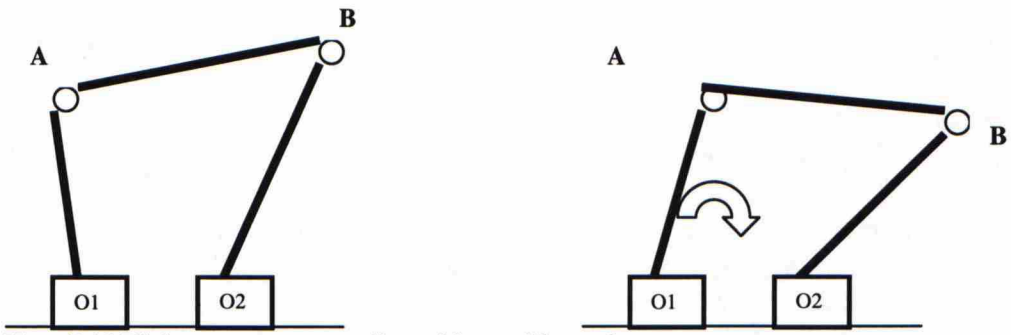
Kuva 2.9 Robotin kohdekoordinaatistot, vasemmalla ylhäällä on maailmankoordinaatisto ja oikealla työkalukoordinaatisto sekä alhaalla nivel- eli peruskoordinaatisto. [4]

2.2 KINEMATIIKKAA JA ROBOTIN GEOMETRISET RIIPPUVUUDET

2.2.1 Johdanto

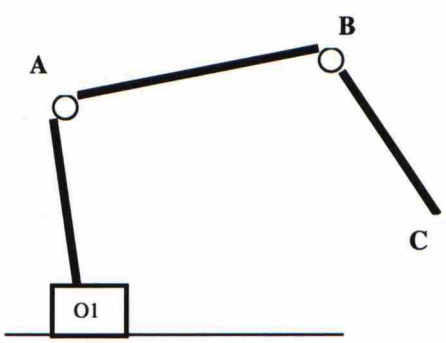
Robotti koostuu useista tukivarsista, joista kaksi liikkuu toistensa suhteen joko jonkin suoran suuntaisesti tai suoran ympäri (kiertoliike). Usein tätä akselia kutsutaan robotin niveleksi. Näiden nivelien avulla tukivarret muuttavat keskinäisiä asentojaan ja asemiaan. Tätä robotin perusliikettä eli siis niveltä kutsutaan robotin vapausasteeksi (DOF, degree of freedom). Nykyisissä roboteissa on yleensä kuusi tai neljä vapausastetta. Yleisin mekaaninen rakenne robotissa on sellainen, jossa tukivarsi aina kytketään edellisen perään (serial linked). Tätä rakennemuotoa kutsutaan avoimeksi

kinemaattiseksi rakenteeksi (kuva 2.11). Tukivarret voidaan myös kytkeä rinnakkain, jolloin rakennetta kutsutaan suljetun kinematiikan rakenteeksi (kuva 2.10).



Kuva 2.10 Yhden vapausasteen suljetun kinematiikan rakenne.

Tällaisessa rakenteessa annettaessa arvo tietylle nivelen kulmamuuttujalle määrätään samalla automaattisesti muidenkin kulmamuuttujien arvot. Normaalisti roboteissa halutaan kuitenkin antaa itsenäisesti jokaiselle nivelelle omat arvonsa, mikä suljetun kinematiikan rakenteissa on mahdotonta. Suljetun kinematiikan robottirakenteet ovatkin harvinaisempia, mutta kuitenkin nykyään on markkinoilla myös näitä rakenteita, kun halutaan erikoisen nopeaa toimintaa tai todella jäykkiä mekaanisia rakenteita.

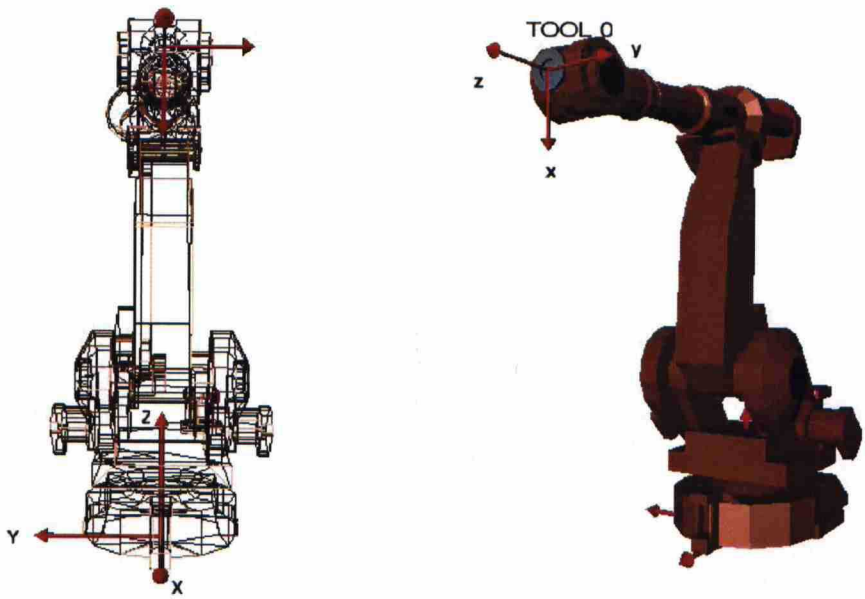


Kuva 2.11 Kolmen vapausasteen avoimen kinematiikan rakenne.

Normaalisti robottien rakenne perustuu siis avoimen kinematiikan rakenteeseen, jossa jokaiselle nivelmuuttujalle voidaan antaa omat arvonsa, mutta käytännössä ei ole kuitenkaan mitään varmuutta siitä, että robotin "end-effector" eli työkalun kärki on tarkoitetussa asemassa.

Robotin ohjausjärjestelmän tärkein tehtävä on hallita työkalunsa asemaa ja liikettä annettujen ohjearvojen mukaisesti. Robotin on siis osattava laskennallisesti muuttaa haluttu työkalun asema robotin oikeiksi vapausasteiden paikkaohjearvoiksi. Tätä tehtävää sanotaan käänteiseksi kinemaattiseksi tehtäväksi (inverse kinematics). Suora kinemaattinen (forward kinematics) tehtävä on työkalun aseman määritys paikka-arvojen perusteella.
Yleisesti käytössä ovat suorakulmaiset ortonormeeratut oikeakätiset koordinaatistot. Maailmakoordinaatisto on robotin työskentely-ympäristöön sidottu robotin ulkopuolinen koordinaatisto. Peruskoordinaatisto on robotin jalustaan sidottu koordinaatisto. Tässä on yleisesti käytetty toteutusta, jossa robotin z-akseli yhtyy

ensimmäisen vapausasteen akseliin ja x-akseli osoittaa ensimmäisen nivelen työalueen keskikohtaan sekä xy-taso yhtyy lattiaan (kuva 2.12).



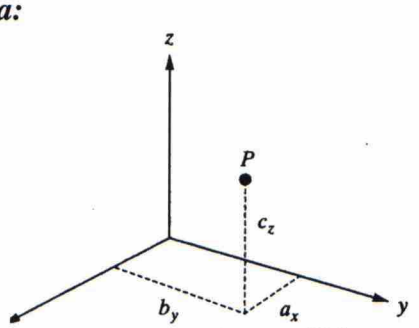
Kuva 2.12 Vasemmalla robotin peruskoordinaatisto ja oikealla kuvattu robotin työkalulaipan (TOOL0) työkalukoordinaatisto, joka on siis suorakulmainen koordinaatisto, johon sidotaan kiinni työkalumäärityksellä haluttu työkalu.

2.2.2 Robotiikan matematiikkaa [4]

Matriiseja voidaan käyttää kuvaamaan pisteitä, vektoreita, kehyksiä, siirroksia, kiertoja ja muunnoksia. Näitä ominaisuuksia hyödynnetään myös robotiikassa.

Pisteen esitysmuoto 3D-avaruudessa:

$$P = a_x\hat{\mathbf{i}} + b_y\hat{\mathbf{j}} + c_z\hat{\mathbf{k}},$$



Kuva 2.13 Pisteen esitys 3-ulotteisessa avaruudessa.

Vektorin esitysmuoto 3-ulotteisessa avaruudessa:
Vektori voidaan määritellä sen alku- ja loppupisteiden koordinaateilla. Esimerkiksi vektorin alkupiste on A ja loppupiste on B, niin saadaan vektori:

$$\overline{P}_{AB} = (B_X - A_X)\hat{i} + (B_Y - A_Y)\hat{j} + (B_Z - A_Z)\hat{k}$$

Vektorin alkupisteen ollessa origossa saadaan vektori:

$$\overline{P} = a_X\hat{i} + b_Y\hat{j} + c_Z\hat{k}$$

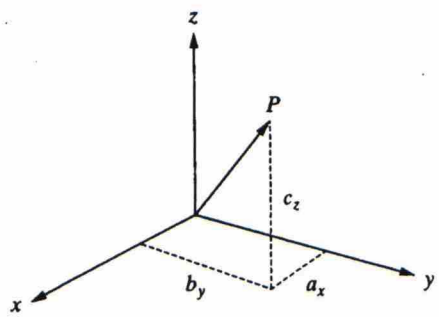
Yleensä aina vektorit esitetään robotiikassa matriisimuodossa:

$$\overline{P} = \begin{bmatrix} a_X \\ b_Y \\ c_Z \end{bmatrix}$$

Tätä esitystä voidaan vielä kehittää ottamalla mukaan skaalauskerroin w, jolloin kyseinen vektori on muodossa:

$$\overline{P} = \begin{bmatrix} X \\ Y \\ Z \\ w \end{bmatrix}$$, jossa $a_x = x/w$, $b_y = y/w$ ja $c_z = z/w$

Muuttuja w voi olla mikä luku tahansa, jolloin sillä voidaan muuttaa vektorin suuruutta. Tietokoneen kuvan zoomaus perustuu juuri tähän esitysmuotoon. Jos luku w on suurempi kuin yksi, niin kaikkia vektorin komponentteja kasvatetaan. Jos se vastaavasti on pienempi kuin yksi, niin niitä pienennetään. Vektorin suuntavektorin esityksessä w = 0, jolloin vektorin pituudella ei ole merkitystä.



Kuva 2.14 Vektorin esitys 3-ulotteisessa avaruudessa.

Esimerkiksi, jos vektorina on $\overline{P} = 3_x\hat{i} + 5_Y\hat{j} + 2_Z\hat{k}$ ja skaalauskerroin w on 2, niin vektorin esitys matriisimuodossa on:

$$\overline{P} = \begin{bmatrix} 6 \\ 10 \\ 4 \\ 2 \end{bmatrix}$$

ja sen suuntavektori matriisimuodossa on vastaavasti:

$$\overline{P} = \begin{bmatrix} 3 \\ 5 \\ 2 \\ 0 \end{bmatrix}.$$

Yksikkövektoria varten on ensin laskettava vastaava yksikkövektorin pituus:

$\lambda = \sqrt{p_X^2 + p_Y^2 + p_Z^2} = 6.16$, jossa $p_X = \frac{3}{6.16} = 0.487$, $p_Y = \frac{5}{6.16} = 0.811$ jne.

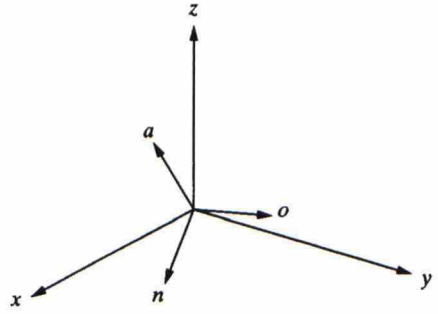
ja yksikkövektori:

$$\overline{P} = \begin{bmatrix} 0.487 \\ 0.811 \\ 0.324 \\ 0 \end{bmatrix}$$

Origoon kiinnitetyn kehyksen esitysmuoto:

Kolmella vektorilla voidaan myös esittää kehys, jonka oma origo yhtyy peruskoordinaatiston origoon. Yleensä nämä kolme vektoria ovat kohtisuorassa toisiaan vasten ja niitä merkitään yksikkövektoreilla $\overline{n}, \overline{o}, \overline{a}$ kuvaamaan normaali-, orientaatio- ja asentovektoreita. Kaikki kolme yksikkövektoria määritellään kolmella komponentilla, joten kehys "Frame" voidaan matriisimuodossa esittää seuraavasti:

$$Frame = \begin{bmatrix} n_X & o_X & a_X \\ n_Y & o_Y & a_Y \\ n_Z & o_Z & a_Z \end{bmatrix}$$

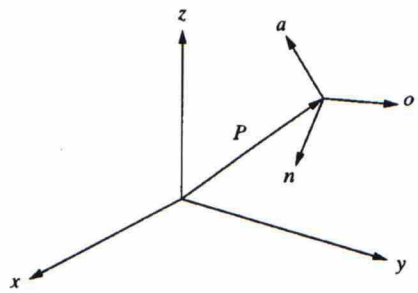


Kuva 2.15 Peruskoordinaatiston origoon kiinnitetty kehys.

Omalla origolla varustetun kehyksen esitysmuoto:

Jos kehys ei yhdy peruskoordinaatiston origoon, niin tällöin ilmoitetaan myös kehyksen origon etäisyys peruskoordinaatiston origosta, minkä tehtävän suorittaa vektori, jonka alkupiste on peruskoordinaatiston origossa ja loppupiste kehyksen origossa (kuva 2.16). Tällöin kehys voidaan määritellä kolmella yksikkövektorilla ja neljäs vektori määrittelee sen sijainnin:

$$Frame = \begin{bmatrix} n_X & o_X & a_X & P_X \\ n_Y & o_Y & a_Y & P_Y \\ n_Z & o_Z & a_Z & P_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

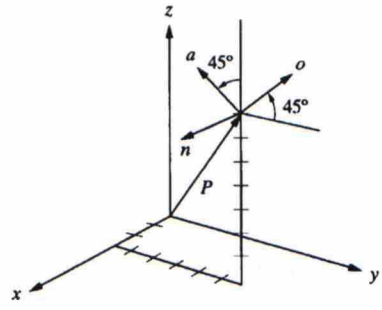


Kuva 2.16 Peruskoordinaatisto suhteessa kehykseen.

Kuten edellisellä sivulla olevasta matriisikuvauksesta voidaan todeta, niin nämä kolme vektoria ovat suuntavektoreita (w=0) kuvaten kolmen yksikkövektorin $\overline{n},\overline{o},\overline{a}$ suuntia ja neljäs vektori (w=1) ilmoittaa kehyksen origon sijainnin peruskoordinaatiston origosta. Tämän neljännen vektorin pituus on tärkeä informaatio ja siksi käytetään skaalauskertoimelle w arvoa yksi.

Esimerkiksi kehyksen "Frame" origo sijaitsee 3,5,7 yksikön etäisyydellä peruskoordinaatiston origosta ja sen n-akseli on yhdenmukainen x-akselin kanssa, o-akseli on 45° kulmassa suhteessa y-akseliin ja a-akseli on 45° kulmassa suhteessa z-akseliin. Tällöin tämä kehys voidaan matriisin avulla määritellä seuraavasti:

$$Frame = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 0.707 & -0.707 & 5 \\ 0 & 0.707 & 0.707 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Kuva 2.17 Esimerkki kehyksen määrittelytä 3D-avaruudessa.

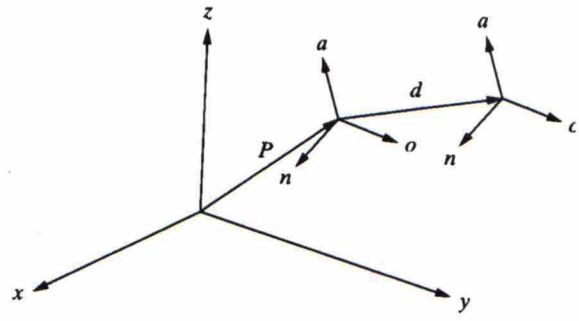
Homogeeninen siirrosmatriisi:
Jos kehyksen siirto avaruudessa tehdään ilman orientaation (kierron) muutosta eli kehyksen suuntavektorit eivät muutu, niin pelkkä siirto voidaan esittää matriisimuodossa seuraavasti:

$$T = \begin{bmatrix} 1 & 0 & 0 & d_X \\ 0 & 1 & 0 & d_Y \\ 0 & 0 & 1 & d_Z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

jossa vektorilla d ilmoitetaan sen komponenttien x, y ja z etäisyyden muutokset alkuperäisestä asemasta.

Lopullinen kehyksen uusi asento saadaan seuraavasti:

$$Frame\ 2 = \begin{bmatrix} 1 & 0 & 0 & d_X \\ 0 & 1 & 0 & d_Y \\ 0 & 0 & 1 & d_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} n_X & o_X & a_X & P_X \\ n_Y & o_Y & a_Y & P_Y \\ n_Z & o_Z & a_Z & P_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_X & o_X & a_X & P_X + d_X \\ n_Y & o_Y & a_Y & P_Y + d_Y \\ n_Z & o_Z & a_Z & P_Z + d_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



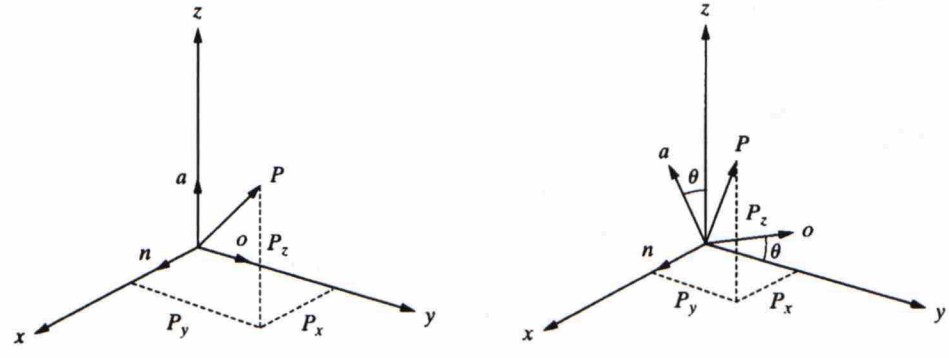
Kuva 2.18 Kehyksen siirto (ei kiertoa) 3D-avaruudessa.

Esimerkiksi siirrettäessä kehystä F yhdeksän yksikköä x-akselin suunnassa ja viisi yksikköä z-akselin suunnassa saadaan sen uusi sijainti seuraavasti:

$$F_{alkup.} = \begin{bmatrix} 0.527 & -0.527 & 0.628 & 14 \\ 0.369 & 0.819 & 0.439 & 3 \\ -0.766 & 0 & 0.643 & 13 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{ja} \quad F_{uusi} = Trans(d_x, d_y, d_z) \times F_{alkup.} \rightarrow$$

$$F = \begin{bmatrix} 1 & 0 & 0 & 9 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0.527 & -0.527 & 0.628 & 5 \\ 0.369 & 0.819 & 0.439 & 3 \\ -0.766 & 0 & 0.643 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.527 & -0.527 & 0.628 & 14 \\ 0.369 & 0.819 & 0.439 & 3 \\ -0.766 & 0 & 0.643 & 13 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

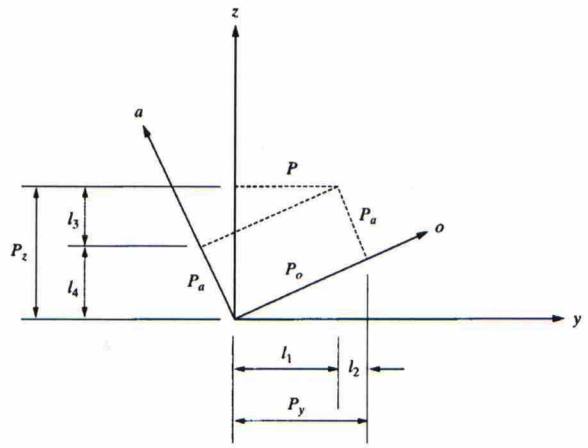
Kiertoliike x-akselin ympäri:



Kuva 2.19 Pisteen P sijainti ennen kiertoa x-akselin ympäri ja sijainti kulman θ verran x-akselin ympäri.

Yllä olevissa kuvissa on esitetty pisteen P kierto x-akselin ympäri kehyksessä, jonka origo yhtyy peruskoordinaatiston origoon ja lisäksi sen akselit $(\overline{n},\overline{o},\overline{a})$ yhtyvät lähtötilanteessa x-, y- ja z-akseleihin. Kiertävä piste P (P_x, P_y ja P_z) kiertyy siis kierrettävän kehyksen mukana.

Alla olevassa kuvassa on tarkasteltu pisteen P koordinaattipisteitä (P_y ja P_z) 2D-tasossa x-akselin suhteen.



Uusiksi koordinaattiarvoiksi pisteelle P saadaan:

$$P_x = P_n$$
$$P_y = l_1 - l_2 = P_o\cos\theta - P_a\sin\theta$$
$$P_z = l_3 + l_4 = P_o\sin\theta + P_a\cos\theta$$

Samat matriisimuodossa:

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} P_n \\ P_o \\ P_a \end{bmatrix}$$

Kuva 2.20 Pisteen P sijainti suhteessa alkuperäiseen kehykseen katsottuna x-akselin suunnasta (2D - näkymä).

Yksinkertaistettu kaavamuoto on $P_{xyz} = Rot(x,\theta) \times P_{noa}$, mutta se esitetään robotiikassa yleensä seuraavasti: ${}^{U}P = {}^{U}T_R * {}^{R}P$, jossa ${}^{U}T_R$ on kehyksen R siirros kehyksen U (universaali) suhteen ja ${}^{R}P$ on P_{noa} (piste P kehyksen R suhteen) sekä ${}^{U}P$ on P_{xyz} (piste P kehyksen U suhteen).

Kiertomatriisi x-akselin ympäri: $$Rot(x,\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

Kiertomatriisi y-akselin ympäri: $$Rot(y,\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

Kiertomatriisi z-akselin ympäri: $$Rot(z,\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Esimerkiksi piste P $(2,3,4)^T$ on kiinnitetty kierrettävään kehykseen, jota kierretään x-akselin suhteen 90°. Pisteen P uudet koordinaattiarvot saadaan seuraavasti:

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} P_n \\ P_o \\ P_a \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ -4 \\ 3 \end{bmatrix}$$

Yhdistettyjen siirrosten ja kiertojen toteutus:
Esimerkiksi tehtäessä samanaikaisesti seuraavat toiminnot:

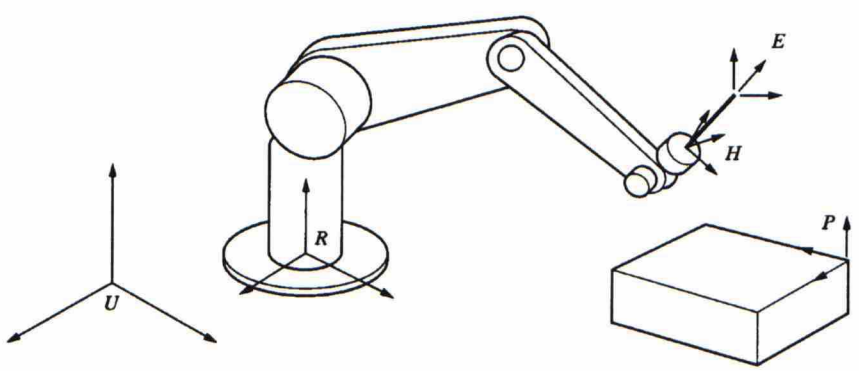
- kierto x-akselin ympäri kulman α verran
  $P_{1,xyz} = Rot(x,\alpha) \times P_{noa}$
- lisäksi siirros [l1, l2, l3] x, y ja z-akselien suhteen
  $P_{2,xyz} = Trans(l_1,l_2,l_3) \times P_{1,xyz} = Trans(l_1,l_2,l_3) \times Rot(x,\alpha) \times P_{noa}$
- näiden lisäksi vielä kierto y-akselin ympäri kulman β verran
  $P_{xyz} = P_{3,xyz} = Rot(y,\beta) \times P_{2,xyz} = Rot(y,\beta) \times Trans(l_1,l_2,l_3) \times Rot(x,\alpha) \times P_{noa}$

Tehtäessä ensin 90° kierto z-akselin ympäri ja seuraavaksi 90° kierto y-akselin ympäri sekä lopuksi siirros [4,-3,7] saadaan lopulliseksi asennoksi:

$$P_{xyz} = Trans(4,-3,7)Rot(y,90)Rot(z,90)\,P_{noa} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 7 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ 10 \\ 1 \end{bmatrix}$$

Siirrosmatriisin käänteismatriisi:
Robotiikan useissa määrittelytehtävissä tarvitaan käänteismatriisia. Alla olevassa (kuva 2.21) kuvassa kuvataan tilannetta, kun halutaan paikoittaa TOOL1(E) kohdekoordinaatistoon P. Esimerkiksi käytännössä halutaan porata reikä kohdekoordinaatistossa P olevaan kappaleeseen työkalulla E. Robotin peruskoordinaatiston suhdetta maailman koordinaatistoon U kuvataan kehyksellä R. Porattavan reiän sijainti maailman koordinaatistoon nähden saadaan:
${}^{U}T_{E} = {}^{U}T_{R} \times {}^{R}T_{H} \times {}^{H}T_{E} = {}^{U}T_{P} \times {}^{P}T_{E}$, josta työkalun E sijainti kohdekoordinaatistossa saadaan siirroksena U→P ja P→E tai vaihtoehtoisesti siirroksina U→R, R→H ja H→E. Todellisessa tilanteessa robotin peruskoordinaatiston R siirros suhteessa maailman koordinaatiston U tiedetään, esimerkiksi asennettaessa robotti tuotantosoluun. Myös työkalukoordinaatiston ${}^{H}T_{E}$ siirros on tiedossa, esimerkiksi määriteltäessä käytettävä työkalu TOOL0:n suhteen. Myös kohdekoordinaatiston ${}^{U}T_{P}$ siirros on tiedossa, esimerkiksi asennettaessa työpöytä maailman koordinaatiston suhteen. ${}^{P}T_{E}$ on myös tiedossa, koska meidän täytyy tietää mihin esimerkiksi kappaleessa reikä porataan.



Kuva 2.21 Maailman (Universal) koordinaatisto, robotin peruskoordinaatisto (R), työkalukoordinaatistot TOOL0(H) ja TOOL1(E) sekä kohdekoordinaatisto(P).

Lopulta ainoa etukäteen tuntematon siirros on ${}^{R}T_{H}$ siis työkalulaipan eli ns. TOOL0:n siirros robotin peruskoordinaatistoon nähden, mikä onkin robottiohjaimen päätehtävä eli sen on laskettava robotin akseleiden asennot saavuttaakseen haluttu loppuasema. Tähän tarvitaan käänteismatriiseja seuraavasti:

$({}^{U}T_{R})^{-1} \times ({}^{U}T_{R} \times {}^{R}T_{H} \times {}^{H}T_{E}) \times ({}^{H}T_{E})^{-1} = ({}^{U}T_{R})^{-1} \times ({}^{U}T_{P} \times {}^{P}T_{E}) \times ({}^{H}T_{E})^{-1}$, mutta koska $({}^{U}T_{R})^{-1} \times ({}^{U}T_{R}) = 1$ ja $({}^{H}T_{E})^{-1} \times ({}^{H}T_{E}) = 1$ saadaan ${}^{R}T_{H}$ seuraavasti:

${}^{R}T_{H} = {}^{U}T_{R}{}^{-1} \times {}^{U}T_{P} \times {}^{P}T_{E} \times {}^{H}T_{E}{}^{-1}$

Esimerkki käänteismatriisin käytöstä, lasketaan rotaatio eli kääntö x-akselin ympäri.

$$Rot(x,\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

Käänteismatriisin määrittelemiseksi tarvitaan:

- laskettava matriisin determinantti
- matriisin transpoosi
- korvataan saadun transpoosimatriisin elementit alideterminanteilla (minor)
- jaetaan saatu matriisi determinantilla

Rotaatiomatriisin transpoosi:

$$Rot(x,\theta)^{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

Lasketaan kaikki alideterminantit (minor-tekijät). Esimerkiksi elementin $_{2,2}$ (cosθ):sta saadaan: cosθ*1 - 0*0= cosθ ja elementti $_{1,1}$ (1):sta saadaan: cosθ* cosθ – sinθ *(-sinθ) =cos²θ + sin²θ=1. Lopputuloksena saadaan: $Rot(x,\theta)^{T}{}_{minor} = Rot(x,\theta)^{T}$. Koska alkuperäisen rotaatiomatriisin determinantti on yksikkömatriisi, niin jakamalla saatu alideterminanttimatriisi determinantilla, niin lopputuloksena käänteismatriisille saadaan:

$$Rot(x,\theta)^{-1} = Rot(x,\theta)^{T}$$

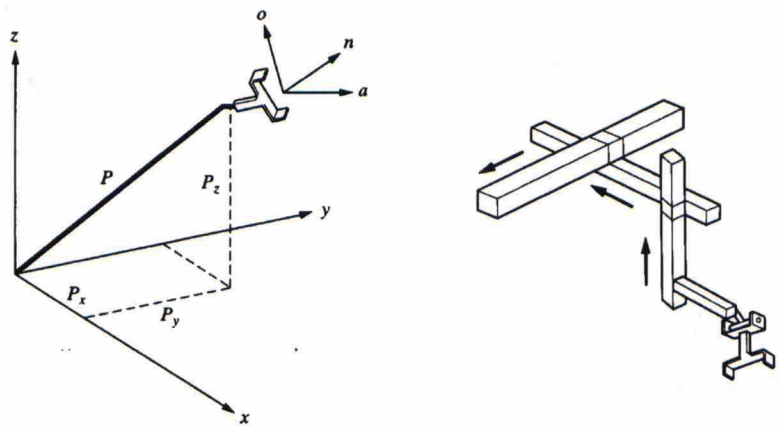
Sama lopputulos saadaan luonnollisesti tehtäessä rotaatio eli kääntö y- tai z-akselin ympäri.

2.2.3 Robotiikan suora- ja käänteinen kinematiikka

Robotin liikkeiden toteutuksessa kinemaattisilla laskuilla voidaan saada vapausasteille tavoitearvoja, joihin vapausasteet tulisi ohjata. Normaalisti tiedetään robotin akselien varsien pituudet ja nivelakselien väliset kulmat eli tiedossa on siis robotin mekaaninen konfiguraatio (Denavit-Hartenberg -parametrit), jolloin robotin työkalulaipan asennon (orientaation) ja aseman (sijainnin) laskutehtävää kutsutaan suoraksi kinemaattiseksi tehtäväksi. Toisin sanoen tiedettäessä robotin vapausasteiden muuttujien arvot saadaan suoralla kinematiikan laskutehtävällä robotin työkalulaipan (end effector) sijainti ja orientaatio. Yleensä tilanne on kuitenkin päinvastainen eli annetaan robotin

työkalulaipalle haluttu orientaatio ja sijainti, jolloin käytetään käänteistä kinemaattista laskutehtävää määritettäessä robotin kunkin vapausasteen muuttujien arvot.

Suora ja käänteinen kinematiikka suorakulmaisessa koordinaatistossa:
Tässä tapauksessa toteutetaan suoraviivaiset liikkeet kolmen pääakselin x-, y- ja z-suunnissa. Tämän tyyppisissä roboteissa kaikki robotin akselien toimilaitteet ovat lineaarisia (hydrauli- ja paineilmasylinterit tai sähkömoottoriruuviyksiköt)



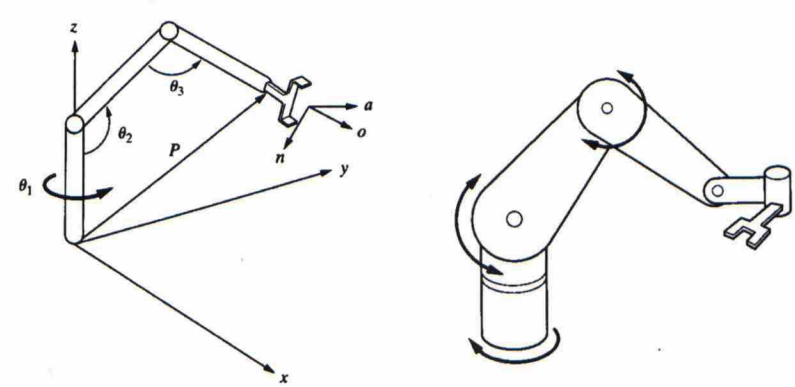
Kuva 2.22 Suorakulmainen eli karteesinen koordinaatisto. [4]

Rotaatioiden puuttuessa tarvitaan siis vain siirroksen siirtomatriisi määriteltäessä uusi paikoitus pisteelle P. Suoran kinematiikan laskutoimitus pisteelle P peruskoordinaatiston suhteen saadaan:

$$ {}^{R}T_{P} = T_{CART} = \begin{bmatrix} 1 & 0 & 0 & P_X \\ 0 & 1 & 0 & P_Y \\ 0 & 0 & 1 & P_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Käänteisessä kinematiikan laskutoimituksessa yksinkertaisesti asetetaan haluttu asema samaksi kuin piste P.

Suora ja käänteinen kinematiikka nivelkoordinaatistossa:
Tämän tyyppisissä roboteissa kaikki robotin akselien toimilaitteet ovat sähköisiä AC- tai DC-servomoottoreita. Erona suorakulmaiseen koordinaatistoon verrattuna on se, että kaikki kolme pääakselia suorittavat kiertoja nivelvarsien välillä.



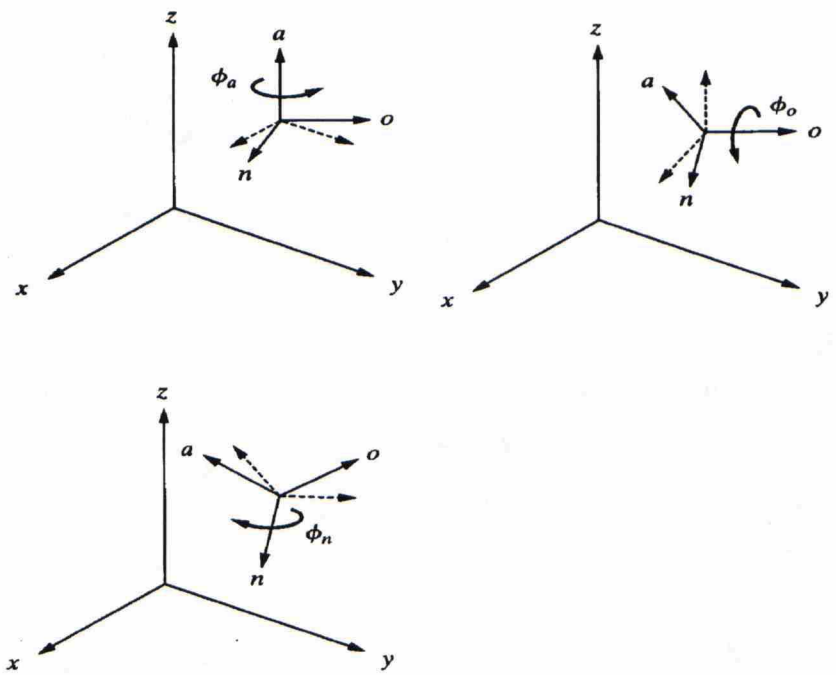
Kuva 2.23 Nivelkoordinaatisto, jota käytetään nivelvarsiroboteissa. [4]

Orientaation eli työkalulaipan (yläkäsivarren tai ranteen) asennon määrittely on oleellinen toimenpide sen jälkeen, kun robotti on liikkunut pääakselien (alakäsivarren) mukaiseen haluttuun asemaan.

Mekaaninen toteutus robotin ranteelle voi olla erilainen:

- Roll, Pitch, Yaw eli RPY -kulmat
- Euler kulmat
- nivelakselit

Roll, Pitch Yaw eli RPY – kulmat:



Kuva 2.24 RPY – rotaatiot $\overline{n},\overline{o},\overline{a}$ -akseleiden suhteen.

- Rotaatiota a-akselin suhteen (liikkuvan kehyksen z-akseli) kutsutaan "Roll".
- Rotaatiota o-akselin suhteen (liikkuvan kehyksen y-akseli) kutsutaan "Pitch".
- Rotaatiota n-akselin suhteen (liikkuvan kehyksen x-akseli) kutsutaan "Yaw".

Esimerkiksi robotin toimiessa pallomaisessa koordinaatistossa saadaan robotin sijainti ja orientaatio seuraavasti: ${}^{R}T_{H} = T_{SPH}(r,\beta,\gamma) \times RPY(\phi_a,\phi_o,\phi_n)$

Euler – kulmat:

Euler kulmien käyttö on hyvin samanlaista kuin RPY -kulmien lukuun ottamatta viimeistä kääntöä, joka tapahtuu myös a-akselin ympäri.

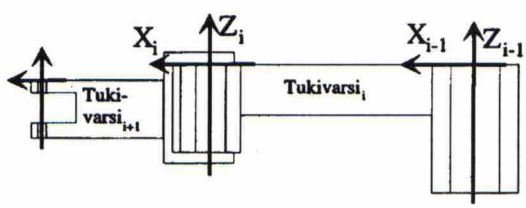
- Rotaatio kulman Φ verran a-akselin suhteen (liikkuvan kehyksen z-akseli)
- Rotaatio kulman θ verran o-akselin suhteen (liikkuvan kehyksen y-akseli)
- Rotaatio kulman ψ verran a-akselin suhteen (liikkuvan kehyksen z-akseli)

Esimerkiksi robotin toimiessa pallomaisessa koordinaatistossa saadaan robotin sijainti ja orientaatio seuraavasti: ${}^{R}T_{H} = T_{SPH}(r,\beta,\gamma) \times Euler(\phi,\theta,\Psi)$

DENAVIT–HARTENBERG – menetelmä:
Kinematiikan perustehtävä on "kiinnittää" kuhunkin tukivarteen ja työkaluun koordinaatisto ja selvittää, miten vierekkäiset koordinaatistot liikkuvat toistensa suhteen, kun robotin nivelet liikkuvat. Tähän on kehitetty 1950–luvulla tekijöiden mukaan nimetty menetelmä ns. Denavit–Hartenberg – menetelmä. Tästä on olemassa kuitenkin kaksi erilaista muotoa. Matemaattinen lopputulos eli homogeeninen siirrosmatriisi on riippuvainen käyttäjän valitsemista koordinaatiston suunnista ja koordinaatistojen sijainnista tukivarren suhteen.

Rajoittavat säännöt koordinaatistojen sijainnille:

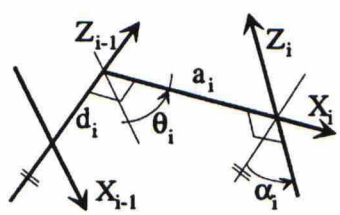
- Z-akseli sijoitetaan nivelen akselin kohdalle tukivarren työkalun puoleiseen päähän.
- Ensimmäiset siirtymät lasketaan robotin peruskoordinaatistosta lähtien.
- X-akseli osoittaa seuraavaa z-akselia kohti.



Kuva 2.26 Esimerkki koordinaatistojen sijoittamisesta tukivarsiin. [1]

Denavit-Hartenberg –parametrit:

- a: kahden perättäisen akselin lyhin mahdollinen etäisyys
- α: kahden perättäisen akselin välinen kulma
- θ: tukivarren kiertymäkulma lähtien akselien lyhimmistä etäisyyksistä
- d: siirtymä nivelen akselin suunnassa



Kuva 2.27 Denavit- Hartenberg – parametrien eräs nimeämistapa.[1]

Denavit-Hartenberg -parametrien ja koordinaatistojen sekä kulmien määrittelyjen jälkeen saadaan homogeeninen siirrosmatriisi, jonka yksi nivel ja siihen liitetty tukivarsi määrittävät.
Siirrosmatriisi:

$$T = \begin{bmatrix} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha & a\cos\alpha \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & a\sin\alpha \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Kulma α on usein suorakulman monikerta, jolloin sen sisältämät termit matriisissa saavat usein arvot -1, 0 tai 1. Samoin kulma θ on yleensä suorakulman monikerta, jollei se ole nivelmuuttuja.

2.3 ROBOTIN TYÖKALUT, SENSORIT JA ANTURIT

2.3.1 Robotin työkalut

Robotin työkalulla tarkoitetaan sitä mekaanista osaa, jota robotti siirtää asemasta toiseen. Työkaluista tavallisin on tarrain. Toinen ryhmä on johonkin prosessiin osallistuvat työkalut, mm. hitsauspistooli, maalausruisku tai liimasuutin. Robottisovelluksessa tarraimen suunnittelu on yksi järjestelmäsuunnittelun välttämättömimpiä asioita.

Tarraimet voidaan jakaa esimerkiksi seuraavanlaisiin ryhmiin:

- avautuvat ja sulkeutuvat tarraimet tarttuvan liikkeen mukaan
- kiertyväsormiset ja rinnakkain suoraviivaisesti liikkuvilla sormilla varustetut tarraimet
- pneumaattiset, hydrauliset tai sähköiset tarraimet
- jäykät ja joustavat tarraimet
- keskittävät tarraimet
- magneettiset tarraimet
- alipainetarraimet
- sisäisesti laajenevat tarraimet
- yksittäinen tarrain, kaksois- tai revolveritarrain
- älykkäät anturoidut tarraimet
- erikoistarraimet

Tarraimen suunnittelu on robottijärjestelmän yksi vaihe, jossa on kaksi nyrkkisääntöä 1) ei saa yrittää matkia ihmisen toimintoja sekä 2) kokonaisuuden miettiminen.
Robotilla ei ole ihmisen monipuolista aistijärjestelmää, eikä robottia ja ihmistä voi verrata työtehtävissä. Tarraimia ja työkaluja suunniteltaessa on katsottava koko automatisointitehtävää kokonaisuutena, jossa tarraimen tai työkalun suunnittelu on vain pieni mutta tärkeä osa.

Työkalun vaihtojärjestelmät mahdollistavat työkalujen ja tarraimien automaattisen vaihdon. Erikseen hankitaan yksi ns. robotin laipan osa ja tarpeen mukaan useampia työkaluihin kiinnitettäviä osia. Yleensä nämä työkalujärjestelmät mahdollistavat myös sähkö- ja ilmaliitäntöjen automaattisen kytkennän. Laadukkaimmat ja samalla kalliimmat järjestelmät varustetaan ns. liukurenkailla (ilma- ja sähkökytkennät), jolloin johdot ja letkut eivät "pyöri" robotin liikkuessa.

Sensoreilla varustettu tarrain:
Tarrain voidaan varustaa omalla ohjauksella, jotta se kykenee mukautumaan ympäristön tai prosessin muutoksiin vastaanottamalla, käsittelemällä ja lähettämällä edelleen tietoa omasta toiminnasta, ympäristöstä sekä muilta laitteilta. Älykkäällä toiminnalla voidaan

estää tuotantokatkoksia, sillä tarraimen välittämän tiedon perusteella tunnistetaan virhetilanteet ja selvitetään ne.
Tarraimen ja robottijärjestelmän suunnittelu vaatii usean tekniikan alan osaamista. Anturoidun tarraimen suunnittelun kannalta keskeisiä osa-alueita ovat tarrainmekaniikka, toimilaitteet, anturit ja ohjausjärjestelmät. Anturien integroiminen tarraimeen ja anturiviestin välitys ohjausyksikköön ovat älykkään toiminnan kannalta hyvin tärkeitä. Tarvittavien ohjaus- ja mittausmenetelmien on oltava tiedossa jo mahdollisimman varhaisessa suunnitteluvaiheessa. Älykkäässä tarraimessa toimilaitetta, mekanismia, työkappaletta ja ympäristöä valvotaan aistimin. Toimilaitteen ja mekanismin asentoa sekä nopeutta mittaamalla saadaan takaisinkytkentätietoa tarraimen sisäisestä tilasta. Ulkoista tilatietoa käytetään ohjamaan tarraimen ja koko robottijärjestelmän toimintaa. [1]
Tarraimen ohjaimena voi toimia robotin ohjausjärjestelmä. Tarraimella voi olla myös erillinen ohjain tai tarrainta voi ohjata solun keskustietokone tai ohjelmoitava logiikka suoraan. Antureiden välittämän tiedon kasvaessa ja monipuolistuessa vaaditaan myös ohjaukselta enemmän. Tarraimen antureilta kerättyä tietoa käytetään tarraimen ja robotin ohjaamiseen. Robottijärjestelmän tilaa valvotaan myös muilla antureilla.

Älykkään tarraimen toiminnot:

- anturien luku ja signaalien käsittely
- tarraimen toiminnan ohjaus
- kommunikointi robotin kanssa
- kommunikointi ympäristön kanssa

Robotissa käytettävät muut työkalut:
Työstävien työkalujen kanssa (esim. hionta, tasoitus ja jäysteenpoisto) kappaleiden muotopoikkeamien aiheuttamaa epätarkkuuden ongelmaa on usein pienennetty joustavien työkalujen avulla. Työkalu joustaa, jos epätarkkuudet aiheuttavat työkalulle voimia. Joustavuuden ollessa sopivasti suunnattu robotin ja ympäristön välinen epätarkkuus ei aiheuta prosessissa ongelmia, vaan joustavuutta käytetään jopa hyväksi jäljen tasoittamiseksi ja robotin anturiohjauksen viiveiden vaikutusten poistamiseksi. Menetelmien käyttökelpoisuutta parantaa niiden edullisuus. Oikeanlainen joustavuus ja anturointi on suunniteltava jokaiselle työkalulle ja työkappaleelle erikseen.

Tavallisia prosessityökaluja:

- kaari- ja pistehitsauspäät
- ruiskumaalaus-, liimaus- ja saumasuutin
- jyrsin tai hiomalaite
- polttoleikkain
- valukauha
- ruuvaustyökalu tai niittauslaite

Konenäköjärjestelmät mahdollistavat kameratekniikan ja tietokoneohjelmistojen käytön avulla hahmon ja kappaleentunnistusta robottisolussa. Konenäköjärjestelmien mahdollisuudet ja hyödyt on tunnettu robotiikan ja tuotantoautomaation sovelluksissa jo pitkään. Ensimmäiset erittäin yksinkertaiset näköjärjestelmät tulivat robotiikkaan 80-luvulla. Sovellukset liittyivät lähinnä kappaleen asennon tunnistamiseen. Suurimman esteen konenäkösovellusten soveltamiselle asetti tietokoneiden liian pieni laskentakapasiteetti. Konenäköön liittyvät laskentatehtävät ovat raskaita ja tarvittava

kapasiteetti saavutettiin vain erikoiskomponenteilla, jolloin järjestelmän kokonaishinta nousi kohtuuttomaksi. Vasta huima kehitys tietokoneiden laskentatehoissa on mahdollistanut edullisten konenäköjärjestelmien toteuttamisen. Konenäköä robottijärjestelmissä tarvitaan silloin, kun perinteinen anturointi ei enää riitä ja kun halutaan minimoida mekaanisten paikoittimien tai kiinnittimien tarve.

Karkeasti näköjärjestelmän tehtävät robottisovelluksissa voidaan jakaa kolmeen ryhmään:

- kappaleen tai kohteen sijainnin määrittäminen eli translaation (x, y z) ja orientaation (Roll, Pitch, Yaw) mittaus käsiteltävästä kohteesta
- luokittelu eli kohteen tunnistus tai luokittelu laadun, muodon, värin, tunnisteiden, koon tai kohteessa sijaitsevan tekstin perusteella
- kohteen mittaus robotin liikeohjelman muokkaamiseksi tai luomiseksi

2.3.2 Sensorit ja anturit robotin työkaluihin

Roboteissa antureita on luonnollisesti sen sisäisessä rakenteessa antamassa tietoja nivelvarsien asennoista ja niiden liikenopeuksista. Seuraavaksi on kuitenkin esitelty mahdollisia ulkoisia antureita, joilla voidaan kommunikoida robotin ja ulkopuolisen ympäristön kanssa ja näin mahdollisesti lisätä adaptiivisuutta robotin ohjauksessa.
Tietoa tarvitaan osien asemasta ja liike- tai pyörimisnopeudesta, mutta myös voimien, paineiden tai lämpötilojen mittaaminen voi olla tarpeen. Tämä tieto kerätään antureilla, jotka mittaavat robotin toiminnan kannalta tarpeellisia fysikaalisia suureita ja muuttavat mittaustulokset robotin ohjausjärjestelmän ymmärtämään muotoon. Useimmiten anturit muuttavat mitattavan prosessisuureen sähköiseksi tai pneumaattiseksi viestiksi. Sähköinen viesti voi olla joko analoginen tai digitaalinen. Analoginen signaali on jatkuva virta- tai jännitesignaali, jonka arvo muuttuu mitattavan suureen muuttuessa. Digitaalinen viesti on yksinkertaisimmillaan lähestymiskytkimissä, joilla on vain kaksi tilaa, virta joko kulkee tai ei kulje. Analoginen anturi voi jakaa 0 ja 1 välisen tilan useampaan osaan, esimerkiksi. anturin antama jännite 0 – 10 V voidaan jakaa 12 bittisellä AD- muuntimella 4096 jako-osaan.

Asema-anturit:
Asema-antureita käytetään mittaamaan kiertymää tai suoraviivaista liikettä. Yksinkertaisimmillaan voidaan käyttää antureita mittaamaan vain läsnäoloa, jolloin käytössä on kaksitilainen anturi. Kaksitilaisia antureita nimitetään myös lähestymis-kytkimiksi, koska ne ilmoittavat työkappaleen tai koneenosan saapumisen määritellylle alueelle. Lähtöviestin jännite- tai virtasignaali on helppo muuntaa ohjausjärjestelmän tarvitsemaksi loogiseksi nollaksi tai ykköseksi. Jatkuvaa asematietoa koneen osien liikkeistä tarvitaan erityisesti säädön toteutuksessa. Koneen osat liikkuvat toistensa suhteen suoraviivaisesti tai kiertymällä (pyörimisliike tai rajoitettu kiertoliike). Näitä molempia suureita voidaan mitata analogia- ja digitaaliantureilla. [3]

Kaksitilaisia antureita robotin yhteydessä voidaan käyttää moniin eri tarkoituksiin, turvarajoina, tunnistamassa siirrettävän kappaleen läsnäoloa, varmistusantureina ja robotin ohjelman yhteydessä käytettävän adaptiivisen toiminnan anturina, jossa esimerkiksi voidaan etsiä kappaletta makasiinista tai mitata kappaleen muotoa.

Mekaaniset rajakytkimet ovat edullisia ja luotettavia antureita, joilla voidaan tunnistaa kappaleiden läsnäolo mm. kuljettimilla, pakkauskoneissa, jne.. Rajakytkin vaatii kappaleen selvän kosketuksen toimiakseen, joten se soveltuu sellaisenaan vain kiinteiden kappaleiden tunnistamiseen. Mekaanisia rajakytkimiä käytetään myös turvarajoina niiden perusluotettavuuden ansiosta. Sallittu kytkentätaajuus on vaatimaton eli vain yhdestä kolmeen kytkentää sekunnissa ja tilanvaihtoon kuluva aika on melko pitkä (2-10 ms), mikä rajoittaa kytkimen käyttöä nopeissa sovelluksissa. Kytkimet soveltuvat vaihtosähkölle (AC) ja tasasähkölle (DC) 250–500 voltin jännitteisiin asti. Sallittu kuormitusvirta (4-16 A) on kerta luokkaa suurempi kuin muilla kaksitilaisilla antureilla.

Induktiivinen kytkin antaa signaalin, kun sähköä johtava materiaali lähestyy anturin tuntopintaa. Induktiivinen kytkin on kosketukseton anturi, mikä antaa paljon sovellusmahdollisuuksia ja takaa mekaanisen kestävyyden. Induktiivinen kytkin tunnistaa luotettavasti vain metalleja. Anturi soveltuu jopa 5000 Hz kytkentätaajuudelle. Käyttöjännitteet ovat yleensä 24–50 VDC tai 250 VAC. Induktiivisien kytkimien käytössä on muistettava tarvittava tunnistusetäisyys. Yleensä se on vain 2mm...20mm, mutta erikoismalleilla jopa 70 mm. Nämä soveltuvat käytettävyyden ja luotettavuuden puolesta hyvin myös robotin tarttujassa käytettäviksi.

Kapasitiivinen kytkin sisältää värähtelypiirin, jonka kapasitanssiin tuntopinnan edessä oleva aine vaikuttaa. Tunnistettavan aineen dielektrisyys muuttaa kapasitanssia ja aiheuttaa tietyssä vaiheessa värähtelijän liipaisun. Kytkentäetäisyys riippuu tunnistettavan materiaalin dielektrisyysvakiosta siten, että aine on vietävä sitä lähemmäksi anturin tuntopintaa, mitä pienempi dielektrisyysvakio on. Tätä ominaisuutta voidaan käyttää hyväksi, jos aine halutaan tunnistaa toisen aineen läpi. Esimerkiksi nestepinnan tarkkailu kapasitiivisella kytkimellä muovi- tai lasiseinän läpi onnistuu, jos nesteen dielektrisyysvakio on selvästi suurempi kuin säiliömateriaalin. Induktiiviseen anturiin verrattuna kapasitiivisella kytkimellä on pienempi kytkentätaajuus (enintään 1000 -1500 Hz), kokoonsa nähden pitempi kytkentäetäisyys (10 -40 mm), laaja materiaalikirjo ja säädettävä herkkyys.

Valosähköisiin kytkimiin kuuluvien optisten kytkimien eli valokytkimien ("valokennojen") etuna on ylivoimainen tunnistusetäisyys (parhaimmillaan 6-8 m), mistä syystä niitä käytetään monipuolisesti kappaleenkäsittelyn automaatiossa ja turvajärjestelmissä. Kytkimeen kuuluu valoa lähettävä diodi ja valoa vastaanottava transistori. Valo on moduloitua, jolloin valokytkimen häiriönsietoisuus paranee etenkin teollisuusympäristön hajavalossa. Rakenteellisesti valokytkimet jaetaan neljään ryhmään:

- vastaanotinperiaatteella toimiva
- lähetin/vastaanotinperiaatteella toimivat
- suoraan heijastavalla periaatteella toimivat
- V-heijastavalla periaatteella toimivat

Robottisovelluksissa viimeisin tyyppi on yleisin, mutta turvavalokennoissa kaksi keskimmäistä tyyppiä ovat ainoat vaihtoehdot. Valokytkimen toimintataajuus vaihtelee alueella 100 Hz-10 kHz. Valokytkimen yleisimmät toimintahäiriöt johtuvat likaisesta ympäristöstä ja runsaasta hajavalosta.

Ultraäänikytkimen suurin tunnistusetäisyys on tyypillisesti 0,2-1 m, enimmillään noin 3 m. Tunnistettavat kohteet voivat olla kiinteää ainetta, nestettä tai jauhetta. Ultraäänikytkimiä voidaan käyttää pölyisissä olosuhteissa kuten hiontasovelluksissa korvaamassa optisia kytkimiä. Tunnistettava materiaali voi olla läpinäkyvä, kiiltävä tai himmeäpintainen. Materiaalin paksuuden tulee olla vähintään 0,01 mm. Kytkintä voi käyttää lähestymiskytkimenä kahdella tavalla, tunnistettava kappale toimii heijastimena tai kappale katkaisee erillisestä heijastimesta saapuvan kaiun. Jos anturilla tunnistetaan jatkuvasti samoja kappaleita, anturi on mahdollista virittää kaiun voimakkuuden perusteella tunnistamaan kappaleita myös eri etäisyyksillä. Ultraäänikytkimen toimintataajuus on 4–15 Hz.

Kiertymää mittaavat asema-anturit antavat jatkuvaa informaatiota kiertymän suuruudesta.
Potentiometri on analoginen absoluuttianturi. Se on perusrakenteeltaan säätövastus, jonka liu'un ja runko-osan välinen resistanssi on yleensä suoraan verrannollinen kiertymään. Monissa kappaleenkäsittelylaitteissa riittää yksinkertainen potentiometri, joka kykenee mittaamaan alle 300–350 asteen kiertymän. Monikierrospotentiometrejä on saatavissa sovelluksiin, joissa tarvitaan usean kierroksen mittausaluetta (5-40 kierrosta). Sallittu liikenopeus on tavallisesti 2000–4000 astetta sekunnissa. Potentiometrin erotustarkkuus on erinomainen ollen tyypillisesti 0,01–0,05 % mittausalueesta. Hyvän potentiometrin lineaarisuus on yleensä 0,1–0,5 %.

Valosähköiset pulssianturit ovat suosittuja digitaaliantureita kohtuullisen hintansa, tarkkuutensa, monipuolisuutensa ja helpon kytkentänsä vuoksi. Niitä on saatavana inkrementti- ja absoluuttiantureina.
Inkrementtianturissa on tavallisesti kiinteä lukulevy ja akselin mukana pyörivä pulssikiekko, jotka ovat noin 0,25 mm etäisyydellä toisistaan. Pulssikiekossa on tarkka kuvioitu vyöhyke, jossa yhtä leveät kirkkaat ja valoa läpäisemättömät sektorit vuorottelevat. Ohjausjärjestelmä tai sen erillinen laskurikortti laskee pulssien nousevia reunoja ja antaa jatkuvan asema- ja suuntatiedon. Inkrementtianturin erotustarkkuus on 360 astetta jaettuna pulssikiekon sektoriparien lukumäärällä.. Kohtuuhintaisilla inkrementtiantureilla mittausepätarkkuus on 0,2–0,5 astetta. Tarkkuuden lisääminen kasvattaa anturin kokoa. Tavallisen pulssianturin halkaisija on 30 - 60 mm ja tarkkuusanturin 80 - 150 mm.. Sallittu pulssitaajuus vaihtelee alueella 10 -100 kHz ja käyttöikä on useita satoja miljoonia kierroksia.

Absoluuttianturi muistuttaa valosähköiseltä rakenteeltaan inkrementtianturia. Absoluuttianturin pulssikiekossa on useita vyöhykkeitä (yleensä 6-12 kpl, tarkkuusantureissa jopa 20 kpl), joissa on vuorotellen valoa läpäiseviä ja läpäisemättömiä sektoreita. Kiekkoa luetaan sädettään pitkin, jolloin ulkokehän kutakin sektoria vastaa yksikäsitteinen binääriluku. Kokonaiset kierrokset lasketaan inkrementtiperiaatteella. Pulssikiekko on koodattu luonnollisella binäärikoodilla, BCD -koodilla (Binary Coded Decimal) tai Gray -koodilla. Absoluuttianturin halkaisija on tarkkuuden mukaan 60–150 mm. Mittausepävarmuus on tyypillisesti 0,2–0,5 astetta. Tarkimmilla antureilla päästään 1-2 kulmasekunnin tarkkuusluokkaan. Sallittu pulssitaajuus on hieman pienempi kuin inkrementtiantureilla.

Synkrot ja resolverit ovat analogisia asema-antureita, joita käytetään erityisesti vaikeissa ympäristöoloissa. Myös robotiikassa resolvereita on käytetty, mutta optiset anturit ovat vallanneet niiltä alaa. Resolverin voi liittää tietokoneohjattuun järjestelmään valmiilla muunninyksiköllä, joka laskee kulman binäärilukuna. Vaihtoehtoisesti lähtösignaalin voi ottaa pulssina. Tulosignaalin ja lähtösignaalin välisestä vaihe-erosta eli pulssin pituudesta lasketaan roottorin kiertymä. Resolverin erottelutarkkuus on samaa luokkaa kuin potentiometrin. Käytännössä kohtuuhintaisten resolverien mittausepävarmuus jää pienemmäksi kuin 0,3 astetta. Parhailla resolvereilla päästään noin 7 kulmasekunnin tarkkuuteen. Näitä molempia antureita on ennen kaikkea käytetty robotin sisäisinä antureina mittaamassa akselien välisiä kulmia.

Lineaariliikkeelle voidaan myös käyttää useita kiertymää mittaavia antureita, valosähköisiä pulssiantureita käytetään yleisesti suoraviivaiselle liikkeelle, joka muunnetaan hammashihnalla anturille sopivaksi pyörimisliikkeeksi. Pulssianturi voidaan tehdä myös suoraan lineaarimalliseksi, samoin potentiometri voidaan tehdä lineaaripotentiometriksi.

Differentiaalimuuntajan eli LVDT -anturin (Linear Variable Differential Transformer) tyypillinen mittausalue on noin 25 mm keskiasennosta molempiin suuntiin. Antureita on olemassa myös hyvin lyhyille liikkeille (1-5 mm) ja pitkille liikkeille (200–600 mm). Differentiaalimuuntaja toimii induktioperiaatteella. Erotustarkkuus on mikrometrin osia. Lineaarisuus vaihtelee välillä 0,2–0,5 % mittausalueesta.

Magnetostriktiivinen anturi perustuu Wiedemann-ilmiöön. Anturin lähtöviesti on joko analoginen virta- tai jänniteviesti tai digitaalinen signaali. Sen siirtoon voi käyttää sarja- tai rinnakkaismuotoista tiedonsiirtoa. Magnetostriktiivisen anturin mittausalue on tyypillisesti 150–2000 mm (25–7600 mm). Erotustarkkuus on 0,01–0,1 % ja lineaarisuus 0,05 % mittausalueesta. Toistettavuus on tavallisesti alle 0,01 mm.

Laser-interferometriaan perustuvassa mittauksessa monokromaattinen säteilijä lähettää valoa puoliläpäisevään peiliin, joka jakaa valon mittaussäteeseen ja vertailusäteeseen. Mittaussäde heijastuu mitattavasta kohteesta ja vertailusäde laitteen sisällä olevasta tasopeilistä. Heijastuneiden säteiden vaiheita verrataan interferometrisesti. Ilmaisimeen syntyy intensiteettiminimi aina kohteen liikuttua valon aallonpituuden puolikkaan verran. Erotustarkkuus on parhaimmillaan noin 200 nm. Menetelmä soveltuu parhaiten pienten siirtymien havainnointiin jopa kymmenien metrien etäisyydellä, jolloin erotustarkkuus on muutaman mikrometrin luokkaa. Kahden pisteen välisen etäisyyden absoluuttisen etäisyyden mittaus sen sijaan on hankalaa.

Laservalopulssin kulkuaikaan perustuvia mittauksia käytetään muutaman metrin mittausetäisyyksillä, jolloin kaupallisten laitteiden mittausepävarmuus on muutaman millimetrin suuruusluokkaa ja erotustarkkuus noin 1 mm. Valopulssin kulkuaika määritetään suorana mittauksena tai vaihe-eromittauksena. Edellisessä menetelmässä puolijohdelaseriin perustuva lähetin lähettää lyhyitä ja voimakkaita jyrkkäreunaisia valopulsseja ja antaa samalla aikamerkin kulkuaikamittaukseen. Vastaanotin havaitsee kohteesta heijastuvan pulssin ja antaa sen perusteella toisen ajoituspulssin. Aikaeron perusteella lasketaan kohteen etäisyys mittalaitteesta. Menetelmä soveltuu myös kohteen nopeuden mittaukseen. Vastaanotettu säde vahvistetaan ja suodatuksen jälkeen sen vaihetta verrataan lähetetyn säteen vaiheeseen. Vaihe-eron perusteella päätellään kohteen etäisyys.

Valonsäteen sirontaan perustuvassa lasermittauksessa mittapään laserdiodi muodostaa mitattavan kohteen pinnalle infrapunavalopisteen. Pinta siroaa valoa ja mittapään valoherkän ilmaisimen pinnalle muodostuu pisteen kuva. Pisteen sijainnin perusteella määritetään mitattavan kohteen etäisyys. Laitteisto soveltuu etäisyyden, pinnan profiilin ja paksuuksien mittaukseen. Menetelmää käytetään mm. sahatavaran lajittelussa ja mittojen valvonnassa, tien pinnan profiilin mittaukseen liikkuvasta ajoneuvosta, auton korin dimensioiden mittaukseen kokoonpanolinjalla ja pursotetun massan paksuudenmittaukseen. Mittausalue on muutamasta senttimetristä puoleen metriin ja mittausepävarmuus noin 0,05 % mittausalueesta. Tässä tutkimustyössä käytetyt laseranturit perustuvat toiminnaltaan juuri valonsäteen sirontaan.
Muita lasermittausmenetelmiä ovat valonsäteen paikan mittaus, kolmiomittaus, intensiteetin muutoksiin perustuva mittaus sekä diodiryhmään ja kamerajärjestelmään perustuva mittaus.

Nopeusanturit:
Nopeuden voi laskea myös asema-anturin lukeman muutoksesta ajan suhteen. Dynaamisessa tilanteessa tämä voi olla kuitenkin liian hidas ja epätarkka menetelmä. Erillisestä nopeusanturista saa suoraan analogisen tai digitaalisen nopeussignaalin.

Takometri on valosähköinen pulssianturi. Rakenteellisesti se on yksikanavainen inkrementtianturi, jonka pulsseista voi laskea kiertymän ilman suuntatietoa. Pulssien lukumäärä aikayksikössä on verrannollinen laskentavälin keskimääräiseen pyörimisnopeuteen. Takometrin suoritusarvot vastaavat valosähköisen inkrementtianturin arvoja.

Sähkömagneettisella pulssianturilla voi mitata helposti ja luotettavasti pyörimisnopeuden. Anturi koostuu käämistä ja kestomagneetista. Mitattavalle akselille kiinnitetään ferromagneettinen kappale, jossa on uloke. Pienellä pyörimisnopeudella käytetään monihampaista pyörijää, jotta mittaustarkkuus saataisiin hyväksi. Hampaan ohittaessa anturin magneettikenttä muuttuu ja indusoi käämiin jännitepulssin, joka voidaan välittää ohjausjärjestelmään sellaisenaan tai muuttaa pulssitaajuuden perusteella analogiajännitteeksi. Pulssianturi on erittäin nopea. Sallittu pulssitaajuus on parhaimmillaan 15–40 kHz. Anturin etäisyyden pyörijän hampaasta tulee yleensä olla alueella 0,5-3 mm. Anturin epälineaarisuus on alle 1 %.

Takogeneraattori on analogia-anturi, jonka lähtöjännite on verrannollinen akselin hetkelliseen pyörimisnopeuteen. Takogeneraattori on useimpien servomoottoreiden vakiovaruste, koska se on nopea ja luotettava anturi moottorin nopeudensäätösilmukkaan. Digitaalisen ohjauksen yleistyminen on vähentänyt takogeneraattorin käyttöä muissa nopeudenmittauskohteissa.

Lineaariliikkeen nopeuden voi mitata ***induktiivisella nopeusanturilla***, joka koostuu käämistä ja kestomagneetista. Anturin sisällä liikkuva magneetti indusoi käämiin liikenopeuteen verrannollisen jännitteen, joka vahvistetaan 4...20 V lähtöjännitteeksi. Epälineaarisuus on yleensä alle 1 %. Suurin liikepituus vaihtelee alueella 100...500 mm. Anturia käytetään mm. robottien siirtonivelien nopeuden mittaukseen.

Voima-, vääntömomentti- ja paineanturit:
Voimien mittauksessa käytetään mm. venymäliuska-antureita, pietsosähköisiä antureita ja induktiivisia antureita.
Venymäliuskat ovat antureita, joiden toiminta perustuu elastisen muodonmuutoksen aiheuttamaan resistanssin muutokseen. Voima-anturiin sijoitetaan yleensä neljä venymäliuskaa siten, että kahteen kohdistuu positiivisia ja kahteen negatiivisia jännityksiä. Venymäliuskat on kytketty ns. Wheatstone - siltaan. Kun anturia kuormitetaan, syntyy jännityksiä, joiden seurauksena anturiin kiinnitettyjen venymäliuskojen vastusarvot muuttuvat. Näin saatu sähköinen viesti on suhteessa anturia kuormittavaan voimaan.
Voima-anturin nimellislähtöjännite on yleensä 2 mV/V. Kun käytetään syöttöjännitettä 15 V, saadaan anturilta viesti 0...30 mV. Viestin vahvistamiseksi käytetään vahvistinyksikköä. Vahvistinyksikössä on yleensä nollapisteen- ja alueensäädöt sekä piirit anturin syöttöä ja kalibrointia varten. Tämän tutkimustyön robotin laippaan sijoitettu voima-anturisovitin sisälsi kolme punnitusanturia, joiden toiminta perustuu venymäliuskojen käyttöön.

Pietsosähköisen voima-anturin toimintaperiaate on varsin yksinkertainen. Pietsosähköistä kidettä puristettaessa siihen syntyy puristavaan voimaan verrannollinen sähkövaraus. Tämä voidaan muuttaa edelleen jännitteeksi varausvahvistimella. Pietsosähköiset anturit on tarkoitettu lähinnä dynaamisiin mittauksiin suurilla taajuuksilla, esimerkiksi iskumaisten ilmiöiden rekisteröintiin. Ne ovat pienikokoisia ja rakenteeltaan jäykkiä. Haittapuolena venymäliuska-antureihin verrattuna on heikompi tarkkuus, huonohko pitkäaikaisstabiilius ja varausvahvistimen tarve.

Vääntömomenttia voidaan mitata kahdella voima-anturilla, jotka ovat laitettu vääntöakselin vastakkaisille puolille. Mitattaessa vääntömomenttia kolmessa suunnassa tarvitaan siis kuusi erillistä voima-anturia. Kuusiakselisen robotin työkalulaippaan kiinnitettävässä voima- ja vääntömomenttisensorissa on kuitenkin yhteensä vain kuusi voima-anturia, koska kolmea voima-anturia voidaan myös käyttää x-, y- ja z-suuntaisten voimien mittaukseen. [4]

Paineen mittaus voidaan periaatteessa toteuttaa kahdella tavalla mittaamalla joko paineen aiheuttamaa voima tai liike. Voima ja liike voidaan edelleen muuttaa sähköiseksi signaaliksi antureilla. Tavallisimpia paineantureita ovat pietsoresistiiviset, induktiiviset ja venymäliuskapaineanturit.

2.4 ROBOTIN OHJELMOINTI

2.4.1 Yleistä

Robottien ohjelmointi alkoi sähkömekaanisista kytkennöistä, joiden avulla saatiin nivelet ajamaan päin haluttuja rajakatkaisijoita vaihe kerrallaan. Myöhemmin roboteille opetettiin käden liikkeitä johdattelemalla eli "nauhoittamalla" nivelten paikka-antureita ja toistamalla näitä liikesarjoja. Suurin osa nykyisistä sovelluksista on tehty siten, että robotille tehdään liikeradat liikuttamalla sen käsivarsi ohjainsauvan avulla tiettyihin asemiin, mutta luomalla toimintalogiikka ja muut lisäoperaatiot tietokoneohjelmalla. Kun robotista ja työympäristöstä on luotu kolmiulotteinen tietokonemalli, niin erillisessä tietokoneessa voidaan tehdä mallipohjaista ohjelmointia. [1]

Ohjelmoinnin keskeisimmät tehtävät:

- Laaditaan toimintajärjestys ja logiikka robottikäsivarren liikkeille sovelluksessa tarvittavien työkalun liikkeiden toteuttamiseksi.
- Tahdistetaan käsivarren liikkeet ympäristön signaaleihin (muut laitteet) tai välitetään muihin laitteisiin tarvittavia tietoja.
- Määritellään robotin toiminta virhetilanteissa.

2.4.2 Johdattamalla ohjelmointi

Käsivarren toimilaitteet vapautettiin ja ihminen liikutti liikkeiden määrittelyvaiheessa lihasvoimin työkalua niin, että haluttu liikerata tuli suoritetuksi. Nivelten paikka-antureiden lukemat tallennettiin liikkeiden aikana instrumenttinauhuriin. Kun liikkeitä toistettiin, nauhuri yhdistettiin nivelten toimilaitteiden säätöpiireihin ohjearvoiksi. Johdattamalla ohjelmoinnista tuli nopeasti vallitseva menetelmä maalausroboteissa. Se mahdollisti maalausrobottien yleistymisen muita sovelluksia nopeammin. Leviämistä auttoi myös maalauksen onnistuminen, vaikka liikeradat eivät toistu aivan tarkasti. Nykyään on saatavilla myös maalausrobotteja, joita ohjelmoidaan samalla tavalla kuin muitakin teollisuusrobotteja. [1]

Johdattamalla ohjelmoinnin vaikeudet:

- Muuttamisen hankaluus, koska yleensä ohjelma täytyy ohjelmoida alusta lähtien uudestaan, kun siihen halutaan muutos.
- Magneettinauhoja oli hankala arkistoida ja käsitellä.
- Ohjelmista on vaikea saada aivan tarkkoja

2.4.3 Opettamalla ohjelmointi

Perinteisesti robotteja ohjelmoidaan viemällä työkalu haluttuun paikkaan käsiohjaimen avulla ja tallentamalla kyseinen asema muistiin. Tätä ohjelmointia käsiohjaimen avulla kutsutaan usein opetukseksi erona päätteeltä tehtävään tekstuaaliseen (eli tekstiä kirjoittavaan) ohjelmointiin, jossa loogisten rakenteiden esittäminen on helpompaa. Näitä ohjelmointitapoja käytetään tavallisimmin yhdessä.

Liikkumista asemien välillä tai kohdeaseman valintaa ohjataan muusta ohjelmoinnista tutuilla menetelmillä eli erilaisilla hyppykäskyillä ja aliohjelmilla.
Robottien ohjelmoinnin välineinä ovat olleet käsiohjaimet, joilla voidaan luoda robottiohjelma kokonaisuudessaan tai ohjaussauva robotin liikuttamiseksi eri interpolaatiotavoilla sekä tavallinen pääte ja editoriohjelmisto. Käsiohjaimen käyttökelpoisuudelle on tärkeää, että robotin käskykantaa voidaan käsiohjaimen kautta selata ilman erillistä ohjekirjaa. Liikeratojen määritystä voidaan nopeuttaa kirjoittamalla uutta robottiohjelmaa päätteen avulla, kun käsivarsi suorittaa toista ohjelmaa, käyttämällä runsaasti suhteellisia asemia, jotka lasketaan muutamista käsivarrella opetetuista asemista sekä kirjoittamalla ohjelmat tekstitiedostoina erillisessä tietokoneessa ja siirtämällä ne robotin ohjausjärjestelmään.

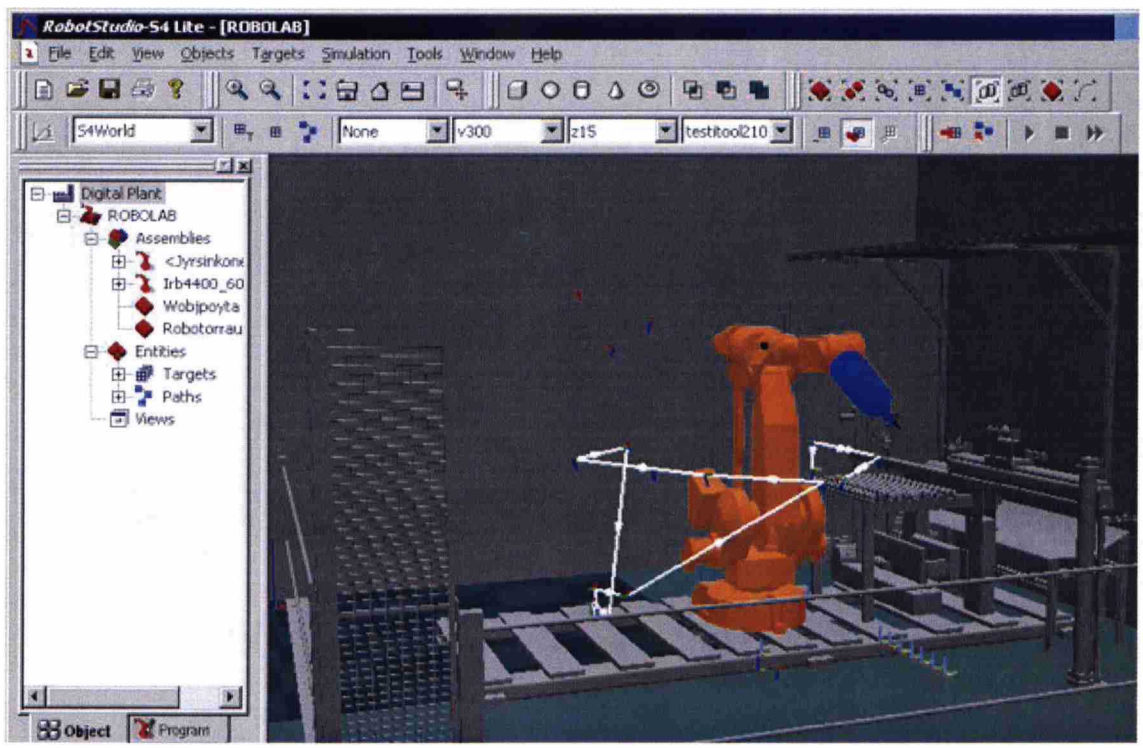
Teollisuudessa robotin käsivarren täytyy tehdä mahdollisimman paljon tuottavaa työtä, siksi pitää liikeratojen määrityksen olla mahdollisimman nopeaa. Pelkästään ulkoisen tietokoneen käyttö ohjelmointiin ei riitä, koska niin robotin kuin työympäristön koordinaatistot ja 3D-mallit ovat kuitenkin jonkin verran epätarkkoja. Tähän eräs ratkaisu on sovelluksen aistinjärjestelmät, jotka huomioivat todellisen maailman epätarkkuudet suunnittelujärjestelmän maailmanmalliin verrattuna. Tällöin voidaan koko ohjelmointiprosessi tehdä ulkoisessa tietokoneessa.
Robottien ohjelmointikielet muistuttivat aluksi Basic-kieltä, johon oli lisätty käsivarren liikekäskyjä. Nykyään ne muistuttavat enemmänkin Pascal-kieltä, mutta globaalisella muuttujavälityksellä. Valitettavasti robotin ohjelmointikielissä on syntynyt vain japanilainen standardi. Jokaisella robottivalmistajalla on siis oma kielensä. Toimittajien vähentyessä ne alkavat tosin muistuttaa toisiaan.

Robotti voi toteuttaa saman liikeradan toisellakin työkalulla, kunhan on tiedossa uuden työkalun ns. työkalukompensointi eli asento- ja paikkaero työkalulaipan ja uuden työkalu koordinaatiston origon välillä. Samoin herkkyys työkalun vääntymiselle törmäysten yhteydessä on pieni, jolloin koko liikerataa ei tarvitse ohjelmoida törmäyksen jälkeen uudestaan, vaan määritellään vääntyneelle työkalulle uusi työkalukompensointi ja myös työkalun kuluminen voidaan kompensoida.
Joidenkin robottien peruskäskykannassa on jopa 150 käskyä.

Kuva 2.28 ABB:n S4C – robottiohjain (ABB:n robottiesitteet).

2.4.4 Etäohjelmointi (off-line)

Mallipohjainen ohjelmointijärjestelmä sijoittuu tuotekehityksen ja valmistuksen väliin. Se voidaan erottaa muista ohjelmointitekniikoista ja myös muista etäohjelmointitekniikoista seuraavan määritelmän mukaan eli off-line tarkoittaa robotin ohjelmointia ilman tuotantorobottia tuotannon ulkopuolisessa tietokoneessa käyttäen 3D-graafista käyttöliittymää, robotin ja sen oheislaitteiden simulointimalleja sekä hyödyntäen valmistettavan tuotteen suunnittelun 3D-muototietoa. [1]



Kuva 2.29 Yleiskuva Lahden ammattikorkeakoulun Tekniikan laitoksen robottilaboratorion 3D-mallista, joka on luotu ABB:n Robot Studio S4 Lite – ohjelmistolla.

Kehittyneimmät mallipohjaiset järjestelmät tukevat eri CAD-suunnittelujärjestelmiä ja useita eri robottimerkkejä. Ne pohjautuvat robottien ja oheislaitteiden simulointimalleihin sekä hyödyntävät tuotemallin muototietoa. Tällaiset järjestelmät ovat samanlaisia kuin CAD/CAM-ohjelmistot numeerisesti ohjatuille jyrsinkoneille ja mahdollistavat ohjelmien verifioinnin eli tarkastamisen etukäteen.

Suunnittelua varten on mallipohjaisissa etäohjelmointiohjelmistoissa:

- 3D CAD -moduuli
- kinemaattisten mekanismien suunnittelumoduuli
- robotti- ja oheislaitekirjastot
- layout-moduuli robottien ja oheislaitteiden sijoitteluun

Robottikirjastossa olevat eri valmistajien robottien simulointimallit vastaavat kinematiikaltaan ja ohjaukseltaan vastaavia teollisuusrobotteja. Laajimmissa ohjelmistoissa voi olla yli 400 eri robottityyppiä yli 50:ltä eri valmistajalta sekä kääntäjät yleisimpien robottimerkkien ohjelmointikielille.

3. ROBOTIN ADAPTIIVINEN OHJAUS

3.1 ROBOTIN OHJAUSJÄRJESTELMÄ

Robotin ohjausjärjestelmän keskeisin tehtävä on ohjata robotin liikkeiden suoritusta halutulla tavalla. Ohjausjärjestelmän toiminnan mukaan voidaan tehdä jaottelu: [5]

- Aidosti pisteohjattu (point-to-point), sekvenssi- eli seurantaohjaus eikä servo-ohjausta ole mukana ollenkaan.
- Aidosti rataohjattu (continuous-path), kaksitasoinen servo-ohjausjärjestelmä, jossa reitti annetaan joko jatkuvina (analoginen) tai tiheästi määräajoin tallennettuina pisteinä (digitaalinen).
- Yhdistetty rata- ja pisteohjaus (computed trajectory), hierarkkinen digitaalinen kaksitasoinen ohjausjärjestelmä (servo-ohjaus), jossa reitti annetaan pisteinä ja pisteiden välisinä vapaina, suoraviivaisina tai kaarevina liikkeinä.
- "Älykäs" robotti (intelligent control)

Ohjausjärjestelmää voidaan myös analysoida sen kehittyneisyyden mukaan:

- sekvenssiohjattu robotti
- opetettava robotti
- yhdistetty piste- ja rataohjattu robotti
- adaptiivinen robotti
- "älykäs" robotti

Yleisimmin roboteissa käytetään akselien ohjauksessa reaaliaikaisia prosessi-tietokoneita, joiden avulla on mahdollista ohjata robotin toimilaitteita tuhansia kertoja sekunnissa ja näin robotin on mahdollista reagoida ympäristön viesteihin millisekunneissa. Reaaliaikaisille ohjausjärjestelmille on ominaista, että niissä voi toimia useita tietokoneohjelmia samanaikaisesti. Tämä on kuitenkin yleensä näennäistä, koska normaalisti kullekin ohjelmalle voidaan varata vain lyhyt ajoaika tietyin aikavälein.

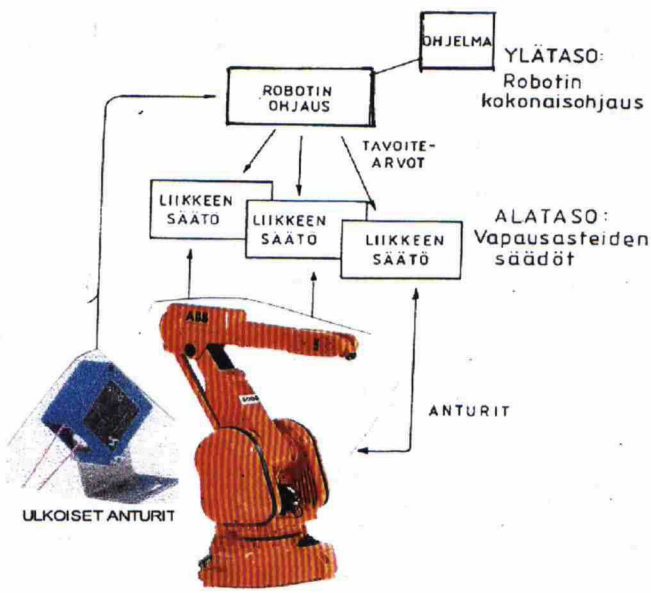
Ohjausjärjestelmän tavallinen koostumus:

- keskusyksikkö
- massamuisti ohjelmien ja parametrien tallennusta varten
- käsiohjelmointilaite eli ns. "TeachPendant"- käyttöliittymä
- ulkoiset liitännät, kuten RS232-C, RS422 ja Ethernet
- robotin akselikohtaiset servo-ohjauskortit
- lisäoptiot, kuten erilaiset väylätekniikat ja vapaat PCI-korttipaikat, esimerkiksi voima-antureiden vahvistin korteille

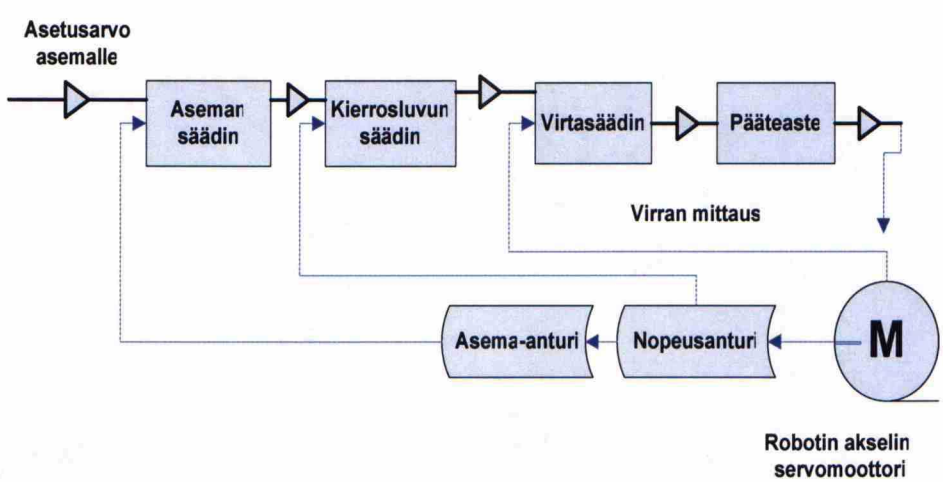
Ohjausjärjestelmän tehtävänä on suorittaa annetut toiminnot liikekäskyiksi ja toteuttaa reaaliajassa myös seurantaa sen suorituksesta eli huolehtia robotin toimilaitteiden takaisinkytkennästä. Tämä on ohjausjärjestelmän vaativin tehtävä, johon liittyy useita eri toimintoja kuten oikea paikkatieto (x, y, z), työkalun oikea asento eli orientaatio (rx, ry, rz), haluttu liikenopeus, kiihtyvyyden ohjaus huomioiden työkalun massa, paikoituksen tarkkuus sekä muut toiminnalliset funktiot. Tämän osalta on viimeaikoina tapahtunut huomattavaa kehitystä tietokoneiden laskentakapasiteetin lisääntyessä. Lisäksi robotinohjausjärjestelmä hoitaa ympäristön havainnoin antureiden avulla.

Ympäristön anturitiedot voidaan yleensä päivittää ohjausyksikköön vain 5–20 Hz:n taajuudella, mikä aiheuttaa ongelmia tietyissä adaptiivisen ohjauksen toteutuksissa. Tässä suhteessa on tapahtunut aivan viime aikoina kehitystä. Ruotsalaisten tutkijoiden [8] tekemissä koeajoissa parhaimmillaan päästiin jo 250 Hz:n taajuuksiin uudemmalla ABB S4C+-robottiohjaimella, johon oli laitettu erillinen lisätietokonekortti varustettuna omalla prosessorilla ja muistilla. Itsediagnostiikka eli robotin sisäisen toiminnan tarkkailu kuuluu myös ohjausjärjestelmän tehtäviin.

Servo-ohjauksen toteutus:
Ei servo-ohjattujen robottien ohjaus toimii kaksitilaisten anturien välittämillä signaaleilla eli liikutaan rajalta rajalle. Servo-ohjauksella toteutettu järjestelmä on huomattavasti monimutkaisempi. Se on yleensä useamman tasoinen hierarkkinen ohjausjärjestelmä.



Kuva 3.1 Servo-ohjatun robotin ohjausjärjestelmän periaate.



Kuva 3.2 Periaatekaavio robotin yhden vapausasteen servosäätöpiiristä.

3.2 ADAPTIIVINEN OHJAUS

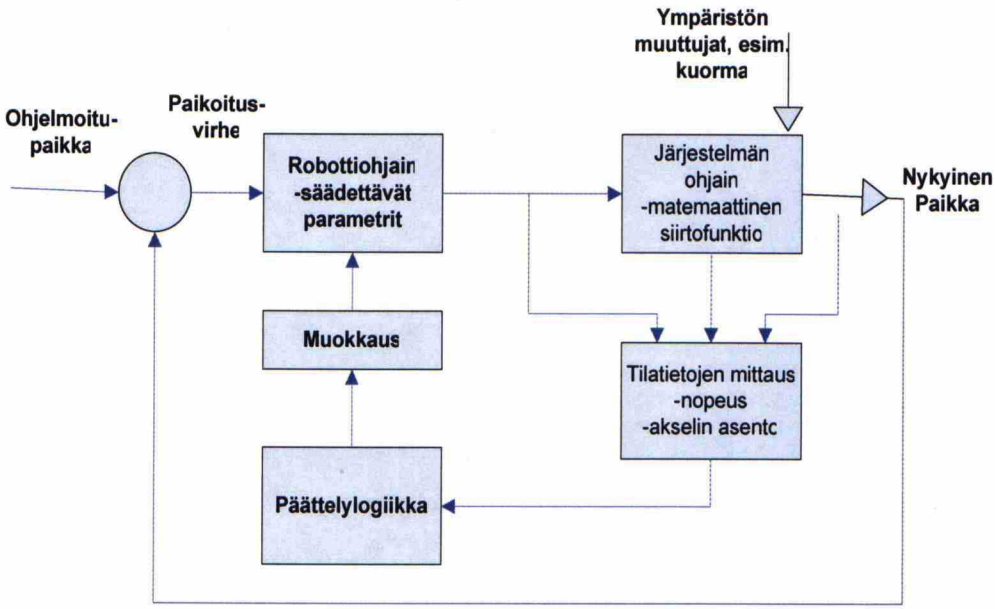
Yleisesti ottaen adaptiivisella säädöllä tarkoitetaan koneissa olevaa mittaus- ja säätöjärjestelmää, jonka avulla saatua mittausinformaatiota käsitellään koneen omassa ohjausjärjestelmässä tai se voidaan myös välittää käsiteltäväksi paremman laskentakyvyn omaavalle yleistietokoneelle. Adaptiivisella eli mukautuvalla säädöllä kone voi itse parantaa omaa toimintaansa eli esimerkiksi robottien yhteydessä parantaa ennalta laskettujen paikoitustietojen arvoja joko numeeriselta tarkkuudeltaan tai paremmin kyseiseen käyttökohteeseen soveltuviksi.

Robottien yhteydessä adaptiivinen ohjaus käsitteenä on laajempi, koska sillä voidaan tarkoittaa varsinaista robotin ohjausjärjestelmän adaptiivisuutta ja ulkoisten antureiden avulla toteutettua adaptiivista ohjausta. Robotin kohdeasemat lasketaan työkalun ja yleensä robotin jalustaan sijoitetun peruskoordinaatiston välisinä etäisyyksinä x-, y- ja z-suunnissa lisättynä työkalun orientaatiotiedoilla. Lisäargumenttina on ainakin tarkkuusparametri eli alue, jonka sisällä robotin haluttu paikoituspiste sijaitsee, esimerkiksi ns. zone -parametrin arvo voi olla 10 mm, jolloin riittää paikoitus-tarkkuudeksi halkaisijaltaan 10 mm ympyrän muodostama alue. Kohdeasemat voidaan määrittää kiinteinä eli pysyvinä arvoina opettamalla ne käsiohjelmointilaitteella tai syöttämällä ne numeerisesti käsin tai off-line-ohjelmiston avulla.

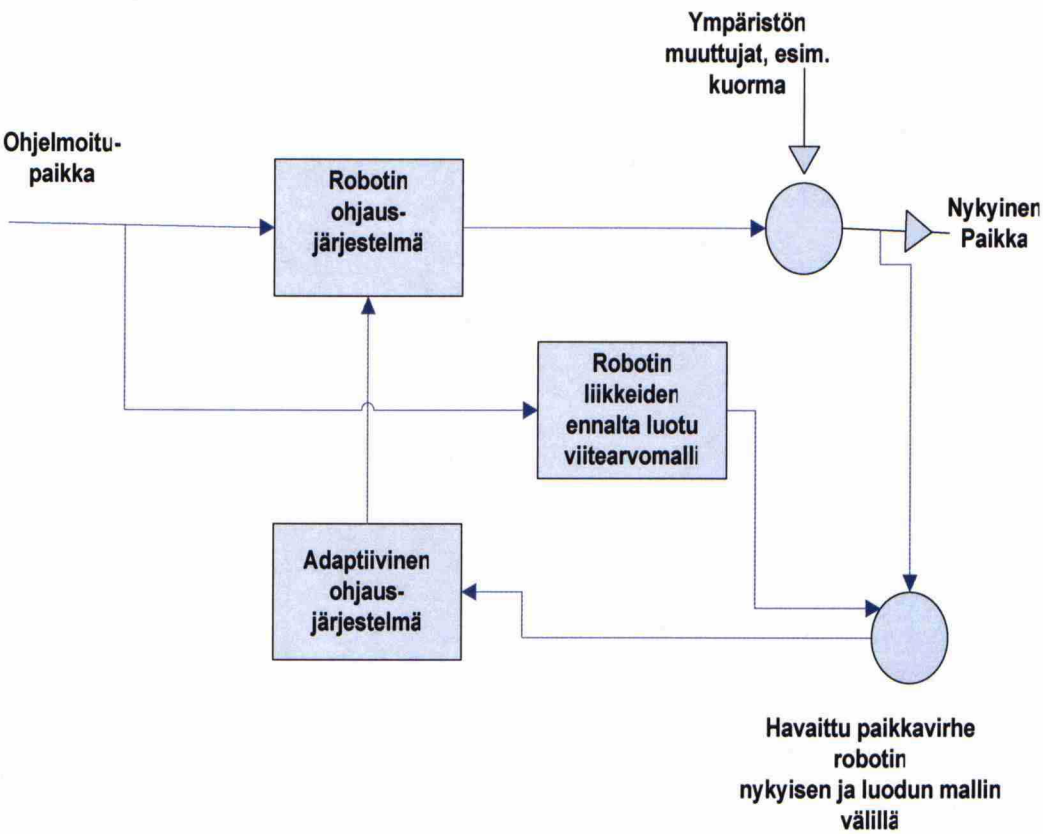
Yksinkertaisimmillaan robottien yhteydessä voidaan puhua "adaptiivisesta ohjauksesta" käyttämällä digitaalista anturia havaitsemaan kohde, jota kohti robotti on ohjelmoitu liikkumaan niin kauan kunnes anturi havaitsee kohteen. Tämä ns. "hakuominaisuus eli searching" on yleensä kaikkien robottivalmistajien käskyvalikoimassa. Tätä ominaisuutta käytetään esimerkiksi noudettaessa tuotteita pinosta, jonka yläpinta tunnistetaan robotin työkalussa olevan anturin avulla. Tätä ominaisuutta voidaan myös käyttää kappaleiden tai robotin ympäristön mittaamiseen.
Monipuolisemmassa muodossa adaptiivinen ohjaus on muodon seurannassa, jossa jollakin analogisella anturilla seurataan kappaleen pintaa ja muodostetaan samalla reaaliaikaista robotin liikerataa ja tätä rataa seuraavalla kerralla voidaan vielä jopa parantaa tallentamalla anturin antamat etäisyysarvot kappaleeseen nähden ja laskennallisesti käsitellä arvoja. Tällöin voidaan jo puhua täysin mukautuvasta ohjauksesta.

Lähes kaikissa nykyisissä robottiohjaimissa käyttäjä voi antaa parametritietoina robotin kantokuorman, halutun paikoitustarkkuuden ja liikenopeuden, jolloin robotti itse optimoi parhaimman suorituskyvyn annettujen parametrien perusteella hakemalla liikkeen suorittamiseksi sopivimmat arvot tietokannasta. Tällaisessa robottiohjaimessa on puutteena kuitenkin se, että käyttäjä ei yleensä voi muuttaa näitä optimointi-arvoja [5]. Tämä puute korjataan ottamalla käyttöön varsinainen robotin adaptiivinen ohjaus (kuva 3.3), jossa näitä robotin tietokannassa olevia suoritusarvoja voidaan muuttaa jatkuvana toimintona takaisinkytkentäantureilta tulevien viestien perusteella. Tällöin usein tulee vastaan kuitenkin muistien tallennuskapasiteetti ja laskennalliset aikarajoitukset, minkä vuoksi on otettu käyttöön työskentelyalueen jako useimpiin osa-alueisiin eli puhutaan ns. zone –määrittelyistä[5].Kuvassa 3.3 on esitetty robotin ohjausjärjestelmä tavallisella adaptiivisella ohjaimella, jossa robotin suorittamaa liikerataa mitataan jatkuvana toimintona ja päättelylogiikka käsittelee saamansa informaation ja päättää sen, kuinka robotin liikerataa kulloinkin on muutettava optimoitaessa paikoitustarkkuutta ja nopeutta.

Lentokoneiden autopilot -ohjausjärjestelmissä käytetään kuvan 3.4 mukaista mallinnettua adaptiivista ohjausjärjestelmää. Oletuksena tällaisessa ohjauksessa on se, että on olemassa juuri oikea vaste halutulle toiminnalle.



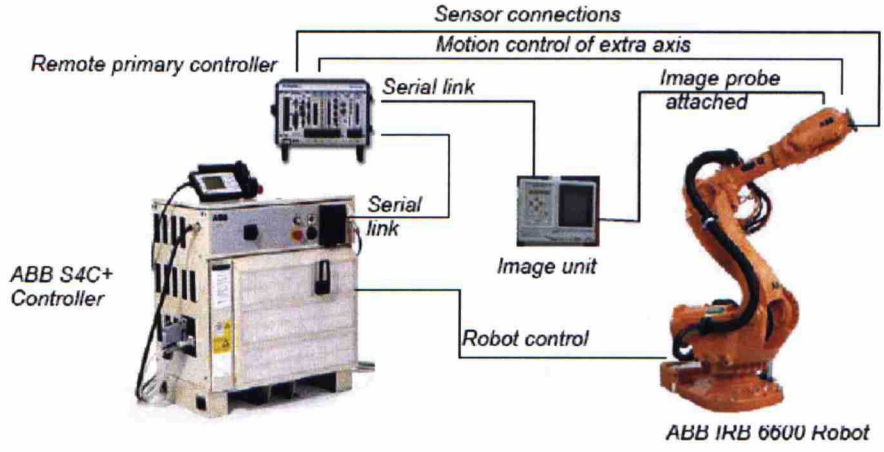
Kuva 3.3 Adaptiivinen robottiohjain, jossa on säädettävät parametrit, lisättynä normaaliin paikoitustakaisinkytkettyyn järjestelmäohjaimeen.



Kuva 3.4 Adaptiivinen robottiohjain, joka perustuu ennalta luotuihin liikearvomalleihin, jolloin saadaan valmiit viitearvot.

3.3 ADAPTIIVISEN OHJAUKSEN SOVELLUKSIA

Teollisuusrobottien ohjaus lihanjalostusteollisuuden sovelluksissa [14]
Australian elintarviketiedelaitos on viime aikoina kehittänyt useita adaptiivisia anturiperustaisia robottisovelluksia lihateollisuuteen. Kaikissa näissä sovelluksissa on robotilta vaadittu nopeaa reagointia sekä on-line tilassa ja off-line kommunikoinnissa. Myös näissä sovelluksissa ilmeni normaalien teollisuusrobottien vajavainen kyky vastata reaaliaikaiseen informaatioon ulkoisilta antureilta. Tässä referenssi-tutkimuksessa on esitelty ratkaisua parantaa tätä kykyä käytettäessä robottia lihan-jalostusteollisuudessa. Lihanjalostusteollisuus on ollut perinteisesti hyvin paljon työvoimaa vaativa ala. Mutta viime aikoina Australiassa on otettu käyttöön kehittyneitä robottijärjestelmiä, esimerkiksi naudan ja sian lihan paloittelusovelluksiin sekä lampaan rasvan erottelutehtäviin. Näissä tehtävissä teollisuusrobotit ovat olleet joustavampia ja tehokkaampia kuin näitä tehtäviä varten valmistetut automaattikoneet ja manipulaattorit.



Kuva 3.5 Lihateollisuuden adaptiivinen robottijärjestelmä.[14]

Yllä olevassa kuvassa on kuvattu järjestelmän laitekokoonpano. Paloiteltavan eläimen ruho havainnoidaan ultraäänianturilla. Ensin ultraäänikuvauksen perusteella tunnistetaan selkäranka, minkä jälkeen ruho halkaistaan vannesahalla, jota robotti liikuttaa nopeudella 75 mm/s ja vaadittavalla 3 mm:n tarkkuudella. Tässäkin tutkimuksessa pullonkaulana oli ohjaimen 10 Hz:n päivitystaajuus ulkoiselta ohjaimelta robottiohjaimelle.

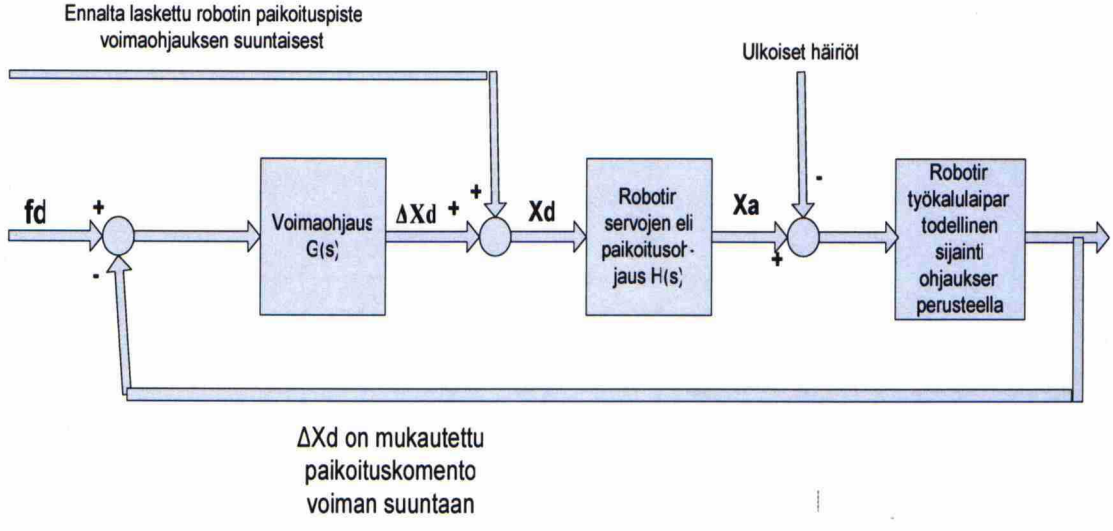
Tämän tutkimuksen tuloksissa ilmeni, että tehokkuutta saadaan teollisuusrobotin toimintaan tämän kaltaisissa sovelluksissa siten, että anturitieto voidaan tavallaan käsitellä ennakolta ennen robotin muita toimintoja. Riippuen ohjelman rakenteesta niin yleensä voi ilmetä melko huomattaviakin viiveitä liikekäskyjen suorituksessa tai työkalun suuntauksessa. Niinpä ohjelman optimointi on hyvin tärkeätä eli käytetään mahdollisimman vähän muuttujia ja rajoitetaan tiedonsiirtopaketin kokoa. Reaaliaikaisen robottiohjauksen parantamiseksi todettiin myös se, että on yritettävä minimoida tarvittavan korjauksen määrä eli toisin sanoen tehdään vain yhden tai kahden robottiakselin liikkeen korjaus mikäli se vain on mahdollista. ABB:n S4C–sarjan robottiohjaimiin saa lisäoptiona moduulin, jossa voidaan määritellä ns. itsenäiset akselit,

joita ovat robottityypistä riippuen esimerkiksi akselit 4 ja 6, jotka voidaan siis määritellä itsenäisiksi, mikä sallii niiden ohjelmoinnin erikseen riippumatta muista robotin akseleista.

Adaptiivinen fuzzy-ohjaus robotin myötäilevän liikkeen ohjaukseen [17]

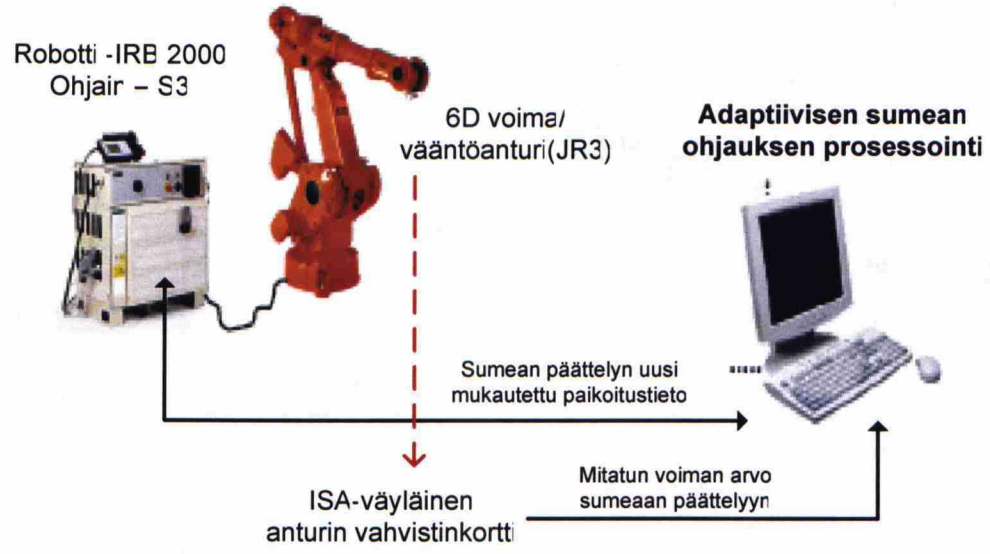
Tässä vertailututkimuksessa fuzzy-ohjain toimii robottiohjaimen rinnalla laskien samanaikaisesti sallittujen voima-arvojen perusteella uudet paikoituspisteet. Tämän toteuttamiseen on tavallaan kaksi mahdollisuutta. Robotin muodostama rata on muodostettu ennalta määriteltyjen pisteiden avulla ja voima on rajoitettu ennalta tiettyjen ympäristöolosuhteiden mukaisesti. Tällainen järjestelmä toimii hyvin yksinkertaisille kappaleille. Toisessa tapauksessa robotin liikenopeus mukautetaan voiman mittauksen mukaisesti, jotta käsivarrelle saadaan kappaleen pinnan mukainen liike. Nyt ei niinkään voiman perusteella muodosteta uutta robotin rataa vaan nopeus asetetaan voiman mukaan.

Robotin paikoitusten ohjaaminen jatkuvan voiman mittauksen perusteella aiheuttaa sen, että saatavat paikoituspisteet eivät ole tarkkoja kuten eivät myöskään mitatut voima-arvot ja tästä aiheutuu käsivarren heilahteleva liike. Näitä rajoitteita ja puutteita on tässä tutkimuksessa yritetty parantaa kehittämällä yhden akselin mukainen adaptiivinen fuzzy-säätöön perustuva voimaohjaus.



Kuva 3.6 Järjestelmän lohkokaaviokuva.

Tutkimuksessa käytettiin ABB IRB 2000 teollisuusrobottia ja JR3 – voima /momenttianturisovitinta robotin työkalulaipassa.



Kuva 3.7 Toteutetun adaptiivisen fuzzy-ohjauksen laitteistokuva.

4. OHJELMOITAVAT LOGIIKAT ja SUMEA SÄÄTÖ

4.1 OHJELMOITAVAT LOGIIKAT

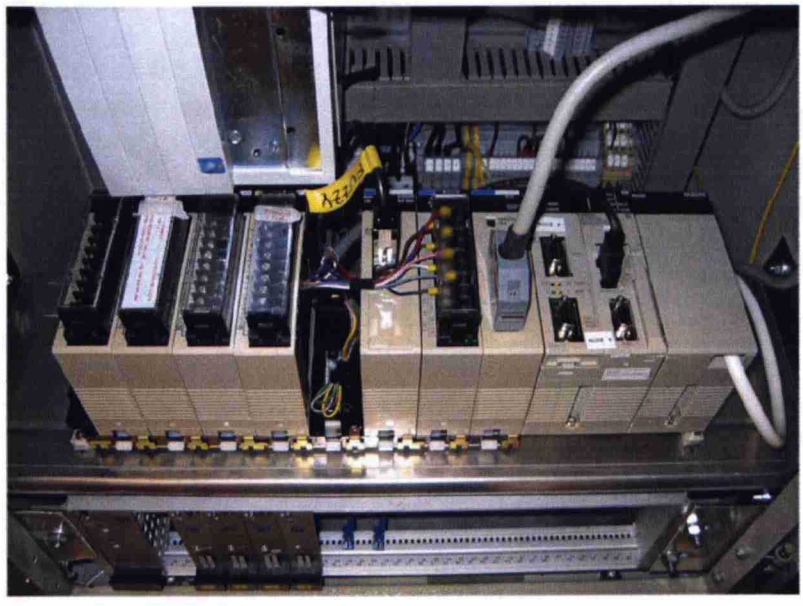
4.1.1 Yleistä

Ohjelmoitavat logiikat (PLC, Programmable Logic Controller) ovat yleisimpiä ohjauslaitteita teollisuuden automaation ohjaustehtävissä. Niiden käskykanta ja suorituskyky on lisääntynyt prosessoreiden kehityksen myötä. Maailman logiikkamarkkinoita hallitsevat monikansalliset yritykset, joista merkittävimpiä ovat Siemens, Mitsubishi, Omron ja Allen Bradley.

Logiikka ohjainlaitteena vastaan ottaa antureilta saamansa informaation ja reagoi saamansa tiedon perusteella ohjelman määräämällä tavalla. Nykyään pienlogiikoiden ominaisuuksiin kuuluu laaja käskykanta, suuri käskyjen käsittelynopeus, askeltavan ohjauksen mahdollisuus, keskeytystulot ja koko logiikan toimintaan vaikuttavien asetusten ohjelmallinen asettaminen. Nykyiset logiikat sisältävät myös sellaisia ominaisuuksia, joihin on aiemmin totuttu vain automaatiojärjestelmien yhteydessä kuten laskentaa, säätöä, videovalvomototeutuksia ja sitä kautta hälytysten käsittelyä sekä raportointia. [28]

Logiikalla voidaan toteuttaa sekä ehto-ohjausta että askeltavaa ohjausta. Ehto-ohjauksesta käytetään myös nimitystä kombinaatiologinen ohjaus, jolla tarkoitetaan

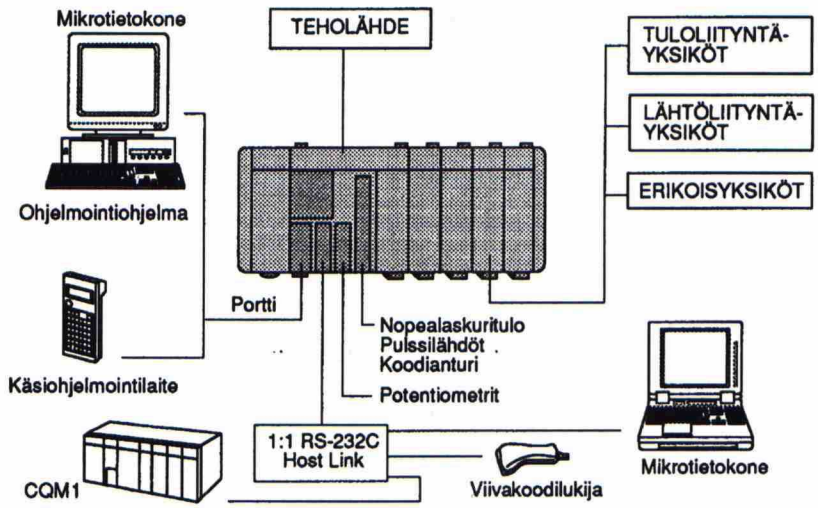
tavanomaista ohjausta, jossa työvaiheet eivät seuraa toistuvasti toisiaan, vaan toimilaitteita ohjataan pelkästään antureilta saadun informaation perusteella. Askeltava ohjaus ottaa huomioon antureilta saamansa informaation lisäksi sen, että ollaan oikeassa askeleessa. Askeltava ohjaus koostuu toisiaan seuraavista askeleista ja siirtoehdoista.



Kuva 4.1 Keskisuuri ohjelmoitava logiikka, OmronCS1G-CPU42H, jossa on ohjelmamuistia 10 000 riviä ja datamuistia 64 000 sanaa (16 bittinen) eli 128 kB. Yhden käskyn suoritusaika on 0.04 µs eli 40 ns, jolloin logiikan prosessorin kellotaajuus on 250 MHz.

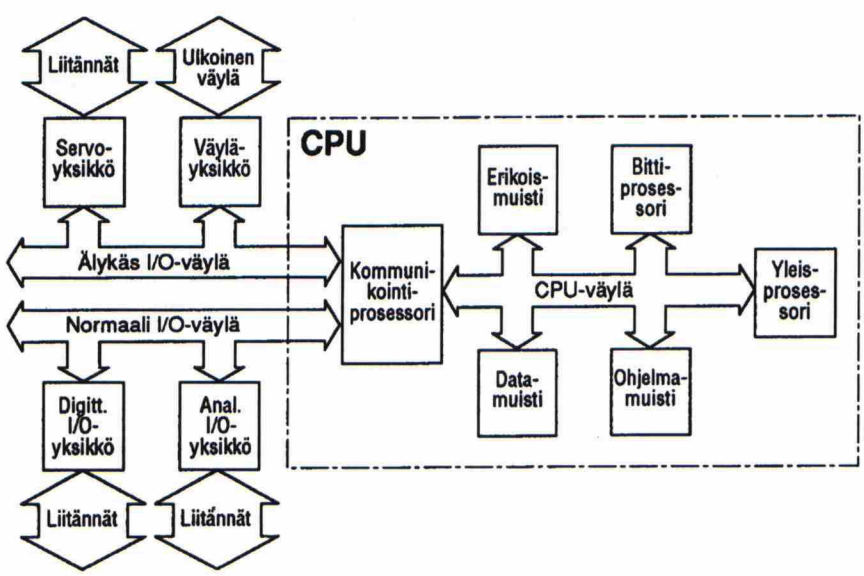
4.1.2 Rakenne

Logiikat jaetaan perinteisesti pieniin kompakteihin ja modulaarisiin logiikoihin. Pienet kompaktit logiikat ovat edullisia, rajallisesti laajennettavia, noin 10...30 tuloa/lähtöä (Input/Output) käsittäviä laitteita. Ne on tarkoitettu yhden pienen koneen ohjaukseen. Peruslaite voi sisältää I/O:n lisäksi laskurituloja, analogiatuloja ja RS-232C-liitännän. Kehityksen suunta on ollut kohti modulaarisia logiikoita, koska myös pienet kompaktit logiikat ovat muuttuneet modulaarisiksi.
Modulaarinen logiikka rakentuu jännitelähdeyksiköstä, prosessoriyksiköstä ja sovellukseen vaadittavasta määrästä erilaisia I/O-yksiköitä. Yksiköt asennetaan korttikehikoihin tai takalevyihin. Logiikassa voi olla useita kehikoita, joista yhteen mahtuu 3...12 yksikköä. Prosessorikehikon lisäksi tarvittavia kehikoita kutsutaan laajennuskehikoiksi. Laajennuskehikot samoin kuin yksittäiset I/O-kortitkin liittyvät prosessoriin I/O-väylän avulla. I/O-väylät ovat rinnakkaismuotoisia väyliä ja niiden leveys vaihtelee 8...32 bittiin. Modulaarista logiikkaa käyttäessä käyttäjällä on mahdollisuus valita logiikan kokoonpano, jolloin logiikkaan voidaan liittää myös erikoisyksiköitä, kuten laskuritulo-, asemointi-, sumean säädön yksiköt sekä Ethernet yksikkö ja muita väyläyksiköitä.



Kuva 4.2 Modulaarisen logiikan rakenne (Automaatiolaitteet, Edita Oy [28])

Perusrakenteeltaan logiikka on mikrotietokone. Keskusyksikkö koostuu prosessorista, muistista ja mahdollisista kommunikointiporteista. Prosessoreita voi logiikassa olla useampia, yhdestä neljään kappaletta. Jokaisella prosessorilla on oma erikoistehtävänsä. Yleisprosessori huolehtii käyttöjärjestelmästä ja sanaoperaatioista, logiikkavalmistajan oma ASIC-piiri huolehtii bittioperaatioista ja kommunikointiprosessori huolehtii keskusyksikön eli CPU:n ulkopuolisesta kommunikoinnista.
Digitaalisella tuloyksiköllä on neljä tehtävää 1) välittää on/off-tietoa antureilta keskusyksikölle, 2) toteuttaa galvaaninen erotus, 3) sovittaa anturijännitteet logiikan jännitteeseen ja 4) suojata logiikkaa häiriöiltä. Lähtöyksikön tehtävänä on välittää tietoa toimilaitteille, toteuttaa galvaaninen erotus sekä sovittaa jännitteet logiikan ja toimilaitteiden käyttöön sopiviksi. Lähtöyksiköiden kytkentätaajuus on vain muutama Hz. Lähtöyksikön kytkimenä voi toimia joko rele, transistori tai triakki, joista rele on yleisin. Lähtöyksiköitä on saatavana 4, 8, 16, 32 ja 64 lähdön kortteina.



Kuva 4.3 Ohjelmoitavan logiikan perusrakenne.(Automaatiolaitteet, Edita Oy [28])

Analogisen signaalin vastaanottamiseen tarvitaan analogista tuloyksikköä. Analoginen tuloyksikkö eli AD-muunnin suorittaa signaalille analogi/digitaali-muunnoksen. Se muuttaa esimerkiksi 0...10 V signaalin 16 bitin digitaalisanaksi. Muunnoksessa käytettyjen bittien määrä vaikuttaa suoraan anturilla saavutettavaan erottelukykyyn. Vastaavasti analogiasignaalilla ohjaamiseen tarvitaan analogialähtöyksikkö eli DA-muunnin. Muuntimissa voidaan kytkimien avulla tai ohjelmallisilla asetuksilla asetella, käytetäänkö virta- vai jänniteviestiä. Yleensä signaalina tulo- ja lähtökorteissa voidaan käyttää yhtä tai useampaa eri standardiviestiä, joita ovat 0...20 mA, 4...20 mA, 1...5 V, 0...10 V ja -10...10 V.
Logiikan muisti jakaantuu RAM (Random Access Memory)-muistiin ja PROM-muistiin. RAM-muistille on luonteenomaista se, että sen sisältö tyhjenee, kun logiikka on jännitteetön. Logiikan muisti voidaan jakaa myös käyttötarkoituksen mukaan. Tällöin puhutaan logiikan I/O-avaruudesta. Logiikan I/O-avaruus jakaantuu erilaisiin muistialueisiin, joiden käyttötarkoitus on erilainen. Muistialueita on erilaisia kuten tulo/lähtö-, apumuisti-, puskuroitu apumuisti-, liikennöintimuisti-, ajastin-, laskuri-muisti-, erikoisapumuisti- ja datamuisti alueet.
Erikoisyksiköt ovat usein niin sanottuja älykkäitä yksiköitä. Yksikköä voidaan kutsua älykkääksi yksiköksi silloin, kun sillä on oma prosessori, kuten esimerkiksi tämän tutkimuksen fuzzy-ohjaimessa on.
Analogiayksiköiden yhteydessä mainittiin, että analogiayksiköitä käytetään säätöjen toteuttamiseen. Niiden avulla voidaan toteuttaa säätö sovellusohjelman avulla. Usein näin toteutetun säädön heikkoutena on hitaus. Tästä johtuen logiikoihin on saatavana erillisiä säätäjäyksiköitä, jotka toteuttavat itsenäisesti säätötehtävän logiikan prosessorin antaman ohjearvon avulla. Tällaisia yksiköitä ovat esimerkiksi PID-säätöyksikkö tai sumean säädön yksikkö.

4.1.3 Logiikan toimintaperiaatteet

Logiikan toiminta perustuu loogisten perustoimintojen toteuttamiseen. Diskreettiä tietoa käsitellessään logiikka reagoi tulotietojen muutokseen. Markkinoilla on toiminta-periaatteeltaan kahdenlaisia eli pyyhkäiseviä ja tosiaikaisia logiikoita. Tosiaikainen logiikka pyrkii toimimaan mahdollisimman reaaliaikaisesti, kun taas pyyhkäisevä logiikka toimii syklisesti. Ne eroavat toisistaan ainoastaan siinä, kuinka ne reagoivat tulotiedon muutokseen. Pyyhkäisevä logiikka toteuttaa sovellusohjelmaa tietyin väliajoin. Pyyhkäisy eli ns. scan-toiminto kestää tyypillisesti muutamia millisekunteja. Pyyhkäistessään logiikka lukee ensimmäiseksi tulojen tilat tulojen käyttömuistiin. Seuraavaksi logiikka suorittaa ohjelman, lukee tulojen tilat niiden käyttömuistista ja asettaa lähtötiedot lähtöjen käyttömuistiin. Kun ohjelma on suoritettu, asettaa logiikka lähtöjen käyttömuistin tilat lähtöyksiköille. Pyyhkäisevän logiikan etuna on se, etteivät tulojen ja lähtöjen tilat muutu ohjelman toteuttamisen aikana. [28]
Tosiaikainen logiikka lukee tulojen tilat suoraan tuloliityntäyksiköistä ja asettaa lähtöjen tilamuutokset välittömästi lähtöliitäntäyksiköihin. Tosiaikainen logiikka reagoi siis nopeammin tulon tilamuutokseen. Tulojen ja lähtöjen tilat voivat muuttua ohjelman läpikäynnin aikana. Nykyisissä logiikoissa on mahdollista käsitellä ohjaukselle kriittisiä tuloja ja lähtöjä tosiaikaisesti ja loppuja pyyhkäisevästi. [28]

4.1.4 Logiikan ohjelmointitavat

Logiikan ohjelmoinnin lähtökohtana on usein ohjauskohteen toiminnoista laadittu toimintakaavio tai sanallinen selvitys halutusta toiminnasta. Näistä muokataan logiikkakaaviot, relekaaviot tai toimintadiagrammit, joiden perusteella varsinainen ohjelma ohjelmointilaitteen avulla logiikalle kirjoitetaan.
Ohjelman kirjoittamiseen on tarjolla monenlaisia ohjelmointityökaluja, joissa on käytettävissä erilaisia ohjelman esitystapoja. Yleisimmin käytettyjä ohjelmointitapoja ovat logiikkakaavio- (Function Block Diagram), relekaavio- (Ladder Diagram) ja sekvenssiohjausohjelmointi sekä käskylista (Instruction List) ja joskus jopa korkean tason ohjelmointikieli (Structured Text). Aivan uusinta tekniikkaa edustaa ns. funktioblokkien käyttö, missä on piirteitä olio-ohjelmoinnista. Tässä tutkimustyössä ohjelmointi suoritettiin Omron Oy:n CX-Programmer ohjelmalla relekaavio-periaatteella.
Yleensä pitkissä ohjelmissa ohjelman rakenne on modulaarinen, mikä tarkoittaa sitä, että ohjelma koostuu ohjelmamoduuleista. Ohjelmamoduulit organisoidaan tehtävien mukaan. Ohjelmamoduulit koostuvat virtapiireistä, jotka on esitetty jollakin edellä mainitulla esitystavalla. Ohjelmassa voi olla ulkoisia (global) ja paikallisia (local) muuttujia. Ulkoiset muuttujat ovat kaikkien ohjelmamoduuleiden käytettävissä ja paikalliset muuttujat ainoastaan siinä ohjelmamoduulissa, missä ne on määritelty.
Ohjelmointilaitteet ovat logiikkakohtaisia. Standardin IEC 1131 tavoitteena on yhdenmukaistaa ohjelmointikäytäntöä, jotta ohjelmointilaitteet ja ohjelmointiohjelmat olisivat eri logiikkavalmistajien logiikoiden kanssa yhteensopivia.
Logiikoiden ohjelmointilaitteet voidaan jakaa kolmeen päätyyppiin pieniin käsiohjelmointilaitteisiin, näyttöpäätteellä varustettuihin ohjelmointilaitteisiin ja mikrotietokoneessa toimiviin ohjelmointiohjelmiin, jollainen on esimerkiksi tässä projektissa käytetty CX-Programmer ohjelma. Mikrotietokoneen käyttöä ohjelmoinnissa puoltaa se, että ohjelma voidaan tehdä off-line -ohjelmointina logiikasta erillään ja toisaalta pienet muutokset voidaan tehdä on-line -ohjelmointina pysäyttämättä logiikan ohjaustoimintaa.

4.2 SUMEA SÄÄTÖ eli FUZZY SÄÄTÖ

4.2.1 Mitä on sumea logiikka

Sumea logiikka ja siihen perustuva säätö on ollut jo pidemmän ajan voimakkaan kiinnostuksen ja tutkimuksen kohteena. Siihen perustuvia menetelmiä on sovellettu lähes kaikissa mahdollisissa kohteissa ja myös robottien ohjauksen tehostamisessa. Ennen kaikkea japanilaiset yritykset käyttävät sumeaa ohjausta kilpailuvälineenä uusissa tuotteissa kuten erityisesti erilaisten kulutusvälineiden eli kameroiden, pesukoneiden, liesien, jne. yhteydessä. Nykyisin sumea tekniikka (Fuzzy Technology) koetaan high-tech:na. Syynä sen käytölle on usein se, että sumeiden ohjausten suunnittelu on helpompaa kuin perinteisten ja se voidaan suunnitella robustimmaksi kuin perinteinen säädin.[2]

Sumea ohjaus soveltuu erityisen hyvin seuraaviin tapauksiin:

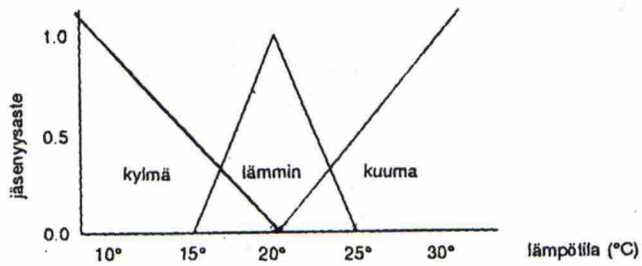
- Ohjattava prosessi on epälineaarinen ja aikariippuva.
- Prosessin malli on vaikea johtaa, jolloin sumea säädin matkii prosessin operaattoria, eikä itse prosessia tarvitse varsinaisesti mallintaa.
- Prosessin käyttäytyminen pystytään ilmaisemaan sanallisesti.

Sumea säätö (Fuzzy Control) on päättelymekanismi, johon prosessin osaava henkilö on laatinut selväkieliset säännöt määräämään sen ohjausta. Sumea säädin voi säätökannan avulla muodostaa täysin epälineaarisia riippuvuuksia tulojen eli mittausten ja lähtöjen eli ohjausten välille. Tällainen säädin voi toimia joko yksimuuttujaympäristössä tai monimuuttujaympäristössä.
Sumeat ohjaukset perustuvat sumeisiin joukkoihin (Fuzzy Sets) ja sumeaan logiikkaan (Fuzzy Logic). Lotfi A. Zadeh esitti ensimmäisenä sumeiden joukkojen teorian vuonna 1965. Zadeh on alun perin iranilainen, mutta hän on toiminut USA:ssa Kalifornian yliopistossa professorina. Hänen esittämien ideoiden pohjalta E.H. Mamdani aloitti sumeiden joukkojen teorian ja sumean logiikan soveltamisen prosessiohjauksiin ja niissä tarvittavaan päätöksen tekoon. Sumea säätö on siis eräs sumean logiikan sovellusalue.[2]

Ennen sumean logiikan teorian kehittämistä nykyaikaiset säätötavat vaativat tarkkoja malleja (joko teoreettisia tai numeerisia) tulojen ja lähtöjen välillä. Matemaattisten kaavojen sijasta sumea logiikka mallintaa tulojen ja lähtöjen suhteita joukolla if/then -lauseita, jotka perustuvat inhimilliseen tietotaitoon ja kokemukseen. Sumean logiikan teoriassa if/then -lauseita kutsutaan säännöiksi. Säännön if -osa kuvaa järjestelmän mahdollisia tiloja ja then -osa kuvaa ehtojen perusteella tehtäviä toimintoja.
Sumean säätösysteemin luominen if/then -lausein on paljon yksinkertaisempaa kuin säädön toteuttaminen perustuen matemaattisiin yhtälöihin. Sääntöjen käyttö ei ole sumean logiikan avainasia, vaan ns. jäsenyysasteen liittäminen näihin sen mukaisesti, kuinka hyvin ne noudattavat edellä mainittuja ehtoja. Kaikki säännöt on yhdistetty niiden jäsenyysasteiden osoittaman tärkeysjärjestyksen mukaisesti. Jäsenyysasteiden liittäminen sääntöihin helpottaa todellisessa prosessissa tapahtuvien muutosten käsittelyä. [2]

Oletetaan esimerkiksi käsite lämmin havainnollistamaan kuinka sumean logiikan teoriassa jäsenyysasteet liitetään ehtoihin. Lämmin on epätarkka käsite. Useimmat ihmiset pitävät 20 °C:n lämpötilaa lämpimänä, mutta pitävät myös 25 °C:n ja 30 °C:n lämpötiloja lämpiminä. Käsite lämmin verrattuna kylmään tai kuumaan ei ole niin tarkka 25 °C:ssa tai 30 °C:ssa. Sumean logiikan teoria liittää jäsenyysasteen käsitteeseen lämmin. Funktiota, joka yhdistää ehdon eli nyt lämmön käsitteen ja jäsenyysasteen, nimitetään jäsenyysfunktioksi. Jäsenyysfunktiolla määriteltyjen eri arvojen joukkoa kutsutaan sumeaksi joukoksi. Tarkin jäsenyysfunktion muoto on yleensä kellomuoto, mutta kolmiota tai trapetsikuviota käytetään useammin niiden helpomman määrittelyn ja käytön vuoksi.

Seuraavassa kuvassa on esitetty kolmion muotoinen jäsenyysfunktio käsitteelle "lämmin".



Kuva 4.4 Jäsenyysfunktiot ja niiden graafiset kuvaajat (Omron, Sumean logiikan käyttöopas).

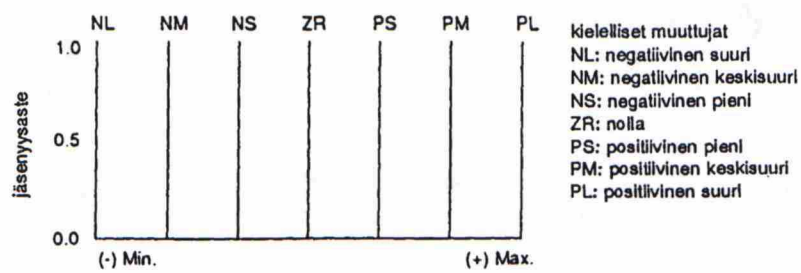
Tästä funktiosta tietokoneessa oleva ohjelma voi laskea "lämpötason" jokaiselle lämpötilalle. Lämpötilan 20 °C jäsenyysaste on 1, kun taas lämpötilojen 17 °C ja 23 °C jäsenyysaste on noin 0,5. Sumean logiikan prosessori laskee jäsenyysasteen jokaiselle säännölle riippuen tiloista ja yhdistää nämä yhdeksi tulokseksi.

If/then -lauseiden luonnissa ensimmäinen vaihe on sumean säätöjärjestelmän luominen eli säädön kuvaaminen if/then -lauseilla. Seuraavat lauseet ilmaisevat tietotaidon, jota ihmiset käyttävät alitajuisesti esimerkiksi säätäessään kylpyvetensä lämpötilaa.

- **If** vesi on hyvin kuumaa **then** käännä vipua paljon vasemmalle
- If vesi on kuumaa then käännä vipua vähän vasemmalle
- If vesi on lämmintä then älä käännä vipua
- If vesi on haaleaa then käännä vipua vähän oikealle
- If vesi on kylmää then käännä vipua paljon oikealle

Sääntöjen if -osia kutsutaan ehto-osiksi, ja then -osia päättelyosiksi ja ne liitetään yhteen loogisilla OR -operaatioilla. Ehtojäsenyysfunktio antaa numeerisen arvon, joka ilmoittaa kuinka hyvin tietyn sumean muuttujan (lämpötila, etäisyys, nopeus jne.) arvo sopii säännön ehto-osaan (kylmä, lämmin, lähellä, kaukana, hidas, nopea jne.).

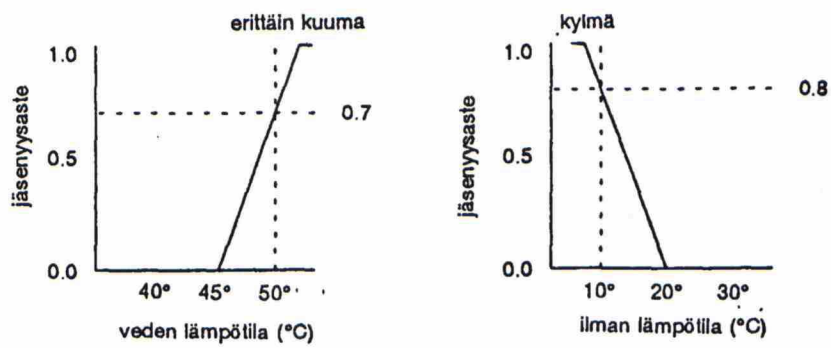
Päättelyosan jäsenyysfunktio määrittelee tarkan arvon toiminnoille (venttiilin kääntäminen, jännitteen nosto jne.), jotka on esitetty säännön päättelyosassa. Päättelyosan jäsenyysfunktion muodolla ei ole juuri merkitystä, joten sitä voidaan likimäärin kuvata pystysuoralla viivalla.



Kuva 4.5 Päättelyosan jäsenyysfunktiot (Omron, Sumean logiikan käyttöopas).

Seuraavaksi täytyy sumean logiikan prosessorin laskea lopputulos jäsenyysfunktion sääntöjen ehto- ja päättelyosista. Ensiksi jokaisen säännön ehto-osan jäsenyysaste lasketaan tulon ja ehtojäsenyysfunktion avulla. Pienin ehto-osan jäsenyysaste on

säännön jäsenyysaste. Minimiarvo valitaan, koska säännön ehto-osat ovat kytketyt loogisilla AND- operaatioilla. Tällöin kaikkien ehtojen on toteuduttava samanaikaisesti. Kun kaikkien sääntöjen jäsenyysasteet on laskettu, lasketaan jäsenyysaste kielellisille muuttujille. Näitä kutsutaan sumeiksi lähdöiksi. Sumea lähtö on kielellisen muuttujan suurin sääntöjäsenyysaste. Maksimiarvo valitaan, koska säännöt on kytketty loogisilla OR -operaatioilla. Maksimiarvon valitsemisella varmistetaan se, että sääntöjen tulokset huomioidaan tasapuolisesti. Lopullinen tulos lasketaan sumeista lähdöistä. Tätä operaatiota kutsutaan selkeytykseksi.
Selkeytyksessä voidaan yleensä käyttää kahta eri selkeytysmenetelmää. Painopistemenetelmä määrittelee painopisteen käyttämällä sumeita lähtöjä "painona" ja kielellisen muuttujan positiota sijaintina. Tässä menetelmässä painopisteen sijainti on lopullinen tulos.
Maksimiarvomenetelmässä käytetään lopputuloksena kielellisen muuttujan sijaintia yhdessä sumeiden lähtöjen maksimiarvojen kanssa. Jos kaksi tai useampi sumea lähtö ovat arvoltaan yhtä suuria kuin maksimiarvo, niin joko vasemmanpuoleisin (pienin) tai oikeanpuoleisin (suurin) valitaan. Ohjelmoitsijan on etukäteen määriteltävä valitaanko vasemman- vai oikeanpuoleisin muuttuja.



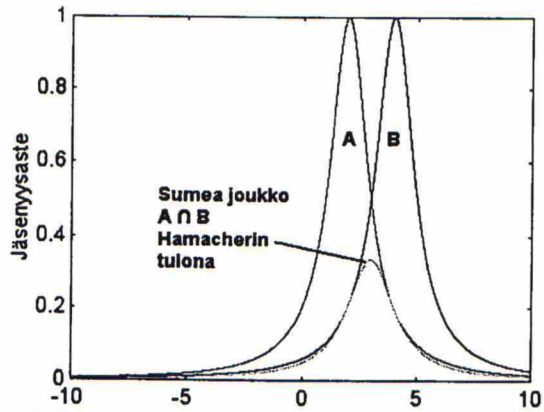
Kuva 4.6 Selkeytykseen valittava jäsenyysaste on pienempi (Omron, Sumean logiikan käyttöopas).

4.2.2 Sumea säätö

Sumeat joukot

Perinteinen logiikka on kaksiarvoisten tilojen tutkimista eli onko jokin tila arvoltaan tosi vai ei. Alkio joko kuuluu joukkoon tai se ei kuulu ja muita vaihtoehtoja ei ole. Sen sijaan sumea logiikka toimii moniarvoisena logiikkana, jossa alkio voi kuulua joukkoon tietyllä jäsenyysasteella, joka on välillä 0...1 oleva luku, joka on puolestaan määritelty jäsenyysfunktiolla. Sumean ohjauksen yhteydessä jäsenyysfunktio (Membership Function) kuvaa muuttujan, kuten paineen, nopeuden, aseman, jne. sumeaan joukkoon kuulumisen astetta. Funktio määritellään siis jatkuvana käyränä, jonka maksimiarvo on 1. Käytännön ohjaustehtavissa jäsenyysfunktion muoto valitaan tarkoituksen mukaan. Valintaan voi vaikuttaa mm. käytettavissä oleva laskentakapasiteetti sekä sumeiden perusjoukkojen avulla määriteltävien muuttujien fysikaalinen luonne. Sumeaan ohjaukseen tai säätöön käytettävien jäsenyysfunktioiden on kuitenkin aina oltava normaaleja ja konvekseja. [2]
Konveksisuusvaatimus merkitsee sitä, että jäsenyysfunktiolla ei voi olla useaita huippupisteitä. Sumeita joukko-operaatioita tarvitaan muodostettaessa uusia sumeita

joukkoja aikaisemmin määriteltyjen joukkojen avulla. Sumeassa säädössä käytetään sumeita joukko-operaatioita mm. yhdistämään kunkin päättelysäännön antamat tulokset lopulliseksi sumeaksi ohjaukseksi.[2]

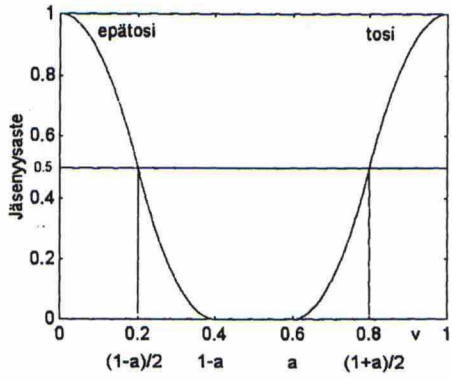


Kuva 4.7 Kahden sumean joukon leikkaus t-normina. [2]

Sumeita loogisia operaatioita voidaan myös määritellä, jos esimerkiksi esitellään kaksi sumeata joukkoa, jotka ovat eri perusjoukoissa, niin näiden sumeiden joukkojen välille voidaan määritellä ns. sumeita loogisia operaatioita. Kahden sumean joukon välille voidaan määritellä loogiset tai-, ja- sekä ei-operaatiot. Sumeille joukoille voidaan myös määritellä algebrallisia operaatioita, kuten summa, erotus ja tulo. Ohjauksen onnistumisen kannalta ei ole suinkaan samantekevää mitä operaatioita käytetään. Yleispäteviä valintaohjeita ei toistaiseksi kuitenkaan ole olemassa. Niinpä mallinnettaessa todellista prosessia sumeiden joukkojen avulla, on tehtävä kokeiluja ja testauksia, jotta löydettäisiin kuhunkin sovellukseen parhaiten toimivat operaatiot. Usein sumean ohjauksen avulla tavoitellaan järjestelmää, joka sopeutuu moneen eri ajotilanteeseen. Tällöin loogisilta operaatioilta edellytetään adaptiivisuutta. Perusoperaatiot min (and) ja max (or) eivät omaa lainkaan adaptiivisuutta, mutta parametroidut operaatiot kuten Hamacher, Yager tai Werner ovat adaptiivisia ja sopeutuvat siten muuttuviin olosuhteisiin paremmin. Monimutkaisissa prosessiohjauksissa loogisten operaatioiden valintaan saattaa vaikuttaa myös käytettävissä oleva säätimen laskentateho. [2]

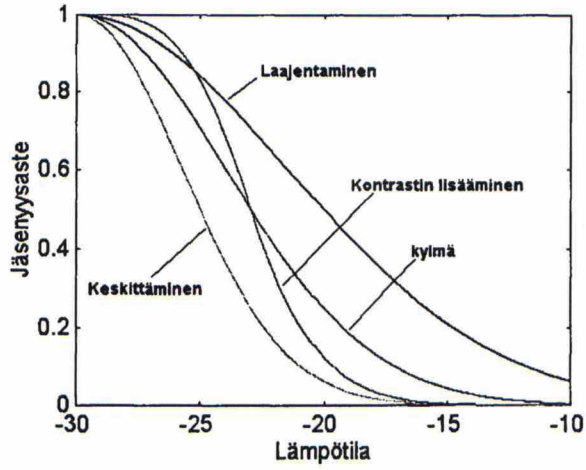
Sumeat joukot tarjoavat perustan monimutkaisen todellisuuden kuvaamiseen. Sumeita joukkoja voidaan käyttää erityisesti kuvaamaan ns. sanallisia muuttujia *(Linguistic Variables).* Sanallinen muuttuja on muuttuja, jonka arvojoukko esitetään sanallisesti. Sanalliset muuttujat ovat havainnollisia ja käyttökelpoisia työkaluja sumean logiikan ja sumean päättelyn yhteydessä. Ne ovat erityisen luontevia ja tuttuja ihmiselle, koska hän käyttää sellaisia päivittäin ja osaa liittää niihin mielekkään merkityksen.

Sumeassa logiikassa esiintyy kaksi erikoisen tärkeätä sanallista muuttujaa, "todennäköisyys" ja "totuusarvo". Totuusarvo voi saada kaksi arvoa, jotka ovat tosi ja epätosi. Kuvassa 4.8 on esimerkki, kuinka parametri a ε [0,1] kuvaa arviota siitä, millä v:n arvolla totuusarvo menee nollaan.



Kuva 4.8 Sanallisen muuttujan totuusarvo ja sen termit tosi ja epätosi. [2]

Alla olevassa kuvassa on esimerkki sanallisen muuttujan muunteluoperaattorien käytöstä. Tavallisimmat muuntelut ovat laajentaminen, keskittäminen ja kontrastin lisääminen.

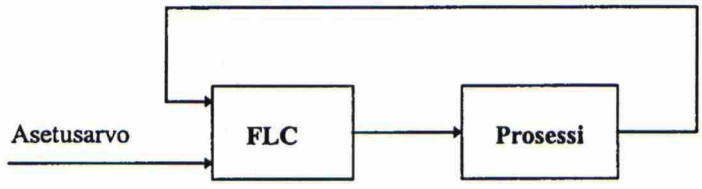


Kuva 4.9 Sanalliseen muuttujaan kylmä kohdistettujen muuntelujen vaikutus. [2]

Sumea säätö

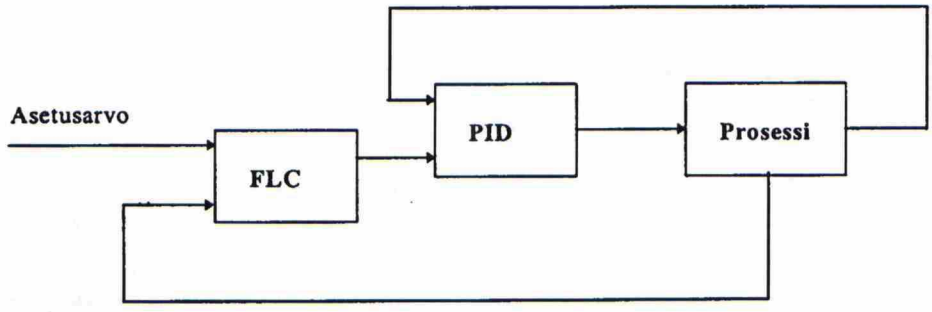
Sumeaa päättelymekanismia voidaan soveltaa säädön yhteydessä monella eri tavalla. Se voi muodostaa yksinään säätöpiirissä tarvittavan ohjauksen. Se voi suorittaa johonkin matemaattiseen säätöalgoritmiin liittyvän osatoiminnon esimerkiksi epälineaarisuuden kompensoinnin, tai se voi suorittaa perinteisen PID -säätimen jatkuvaa viritystä. Edelleen sumeaa päättelyä voidaan yhdistää ns. neuraalilaskennalla toteutettuihin sovelluksiin.

Sumea säädin primäärisäätimenä:



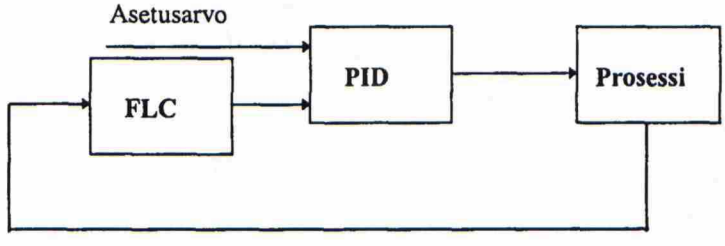
Sumea säädin saa mittaustietoa prosessista ja asetusarvon eli ohjearvon käyttäjältä sekä huolehtii yksinään prosessin säätämisestä.

Sumea säädin ylemmän tason säätimenä:



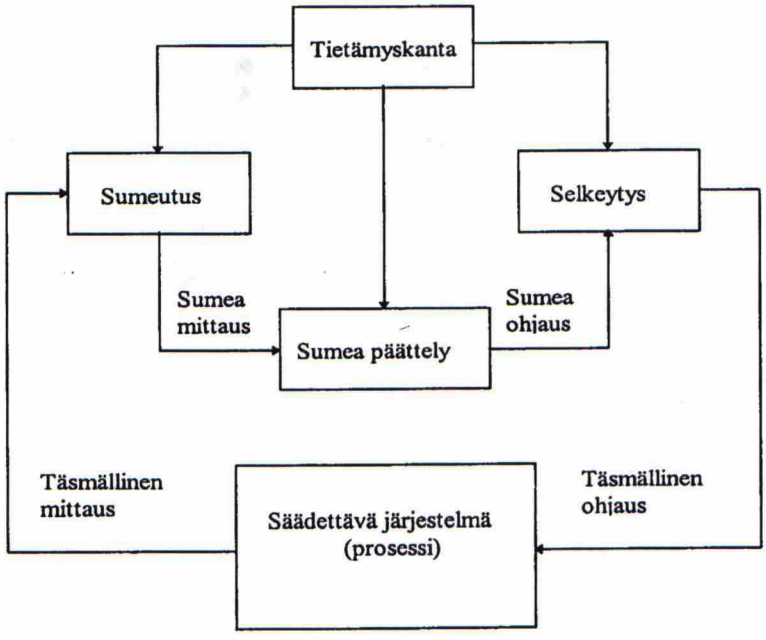
Sumea säädin saa mittaustietoa prosessista ja asetusarvon käyttäjältä sekä määrittää PID–säätimelle kussakin tilanteessa sopivan asetusarvon. PID -säädin puolestaan huolehtii varsinaisesta prosessin säätämisestä.

Sumea säädin mittauksen muodostajan:



Sumea säädin muodostaa prosessista saatavan mittaustiedon perusteella PID- säätimelle tuotavan mittausarvon. Sumean päättelyn avulla voidaan tällöin toteuttaa esimerkiksi erilaisia mittausarvojen lineaarisointikäsittelyjä ja PID- säädin huolehtii varsinaisesta säätämisestä.

Alla olevassa kaaviossa on esitetty sumean säätimen lohkokaavio. Säädin koostuu sumeutuksesta (Fuzzification), tietämyskannasta (Knowledge Base), sumeasta päättelystä (Fuzzy Inference) ja selkeytyksestä (Defuzzification).



Sumeutus (Fuzzification):

- Suoritetaan tulosuureiden eli järjestelmän tilasuureiden mittaus.
- Suoritetaan tulosuureiden mittausarvojen kuvaaminen ja skaalaus sumeiden perusjoukkojen arvoalueille.
- Toteutetaan tulosuureiden mittausarvojen sumeutus sopivilla sanallisilla muuttujilla nimettävien sumeiden perusjoukkojen jäsenyysasteiksi, jolloin syntyy sumeutettu mittaus.

Tietämyskanta (Knowledge Base)

- Tallentaa säätöön tarvittavan tietämyksen käytettäväksi sumean päättelyn ja selkeytyksen yhteydessä.
- Sisältää sanallisen sumean säädön sääntökannan (Linguistic Fuzzy Control Rule Base) sekä mahdollisen tietokannan (Data Base).

Sumea päättely (Fuzzy Inference)

- Muodostaa sumean mittauksen ja tietämyskannan sisällön perusteella sumean ohjauksen käyttäen sumean logiikan päättelysääntöjä.

Selkeytys (Defuzzification)

- Muodostaa sumean ohjauksen ja tietämyskannan sisällön perusteella tarkan ohjauksen, tämä muodostuu kaikista niistä täsmällisistä ohjaussignaaleista, jotka viedään prosessin toimilaitteille ja joilla vaikutetaan prosessin tilaan.

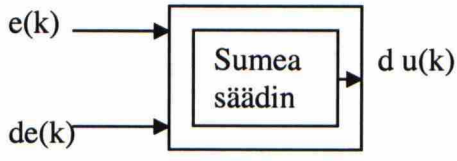
Mittauksen sumeuttaminen:

Sumeassa säädössä suoritetaan prosessin tilasuureiden mittauksia samaan tapaan kuin perinteisessä säädössäkin. Jokaiselle mittaussuureelle määritellään yksi tai useampia

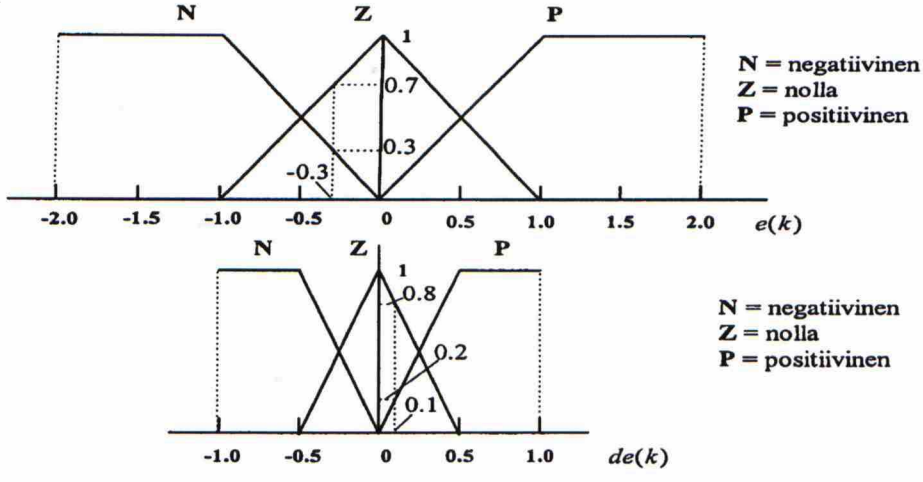
sumeita joukkoja, joihin mittaussuureen arvot voivat kuulua. Nämä sumeat joukot nimetään ja niille määritellään jäsenyysfunktiot. Mittauksen sumeuttaminen tarkoittaa sitä, että lasketaan kunkin mittaussuureen yksittäisten arvojen jäsenyysasteet kyseessä olevalle suureelle sumeissa joukoissa. [2]

Esimerkki

Olkoon PI- tyyppinen sumea säädin, jonka tulosuureet ovat erosuure a ja erosuureen muutos $\Delta e = de$ ja lähtösuureena on ohjausmuutos $\Delta u = du$.



Kummallekin tulosuureelle määritellään kolme sumeaa joukkoa N, Z ja P, joiden jäsenyysfunktiot nähdään alla olevassa kuvassa.



Kuva 4.10 Sumean säätimen jäsenyysfunktiot.[2]

Kukin jäsenyysfunktio voidaan määritellä myös joukolla pisteitä, jotka yhdistyvät toisiinsa suorilla viivoilla. Ne voidaan esittää nyt matriiseina, erosuureen e jäsenyysfunktiot matriisina *mfe* ja erosuureen muutoksen *de* jäsenyysfunktiot matriisina *mfde* avulla seuraavasti:

$$\mathbf{mfe} = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \quad \mathbf{mfde} = \begin{bmatrix} -1 & -0.5 & 0 & 0.5 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

missä matriisin ensimmäinen vaakarivi määrittelee kaikkien jäsenyysfunktioiden nurkkapisteiden x-koordinaatit, toinen vaakarivi määrittelee jäsenyysfunktion N vastaavat y-koordinaatit, kolmas vaakarivi puolestaan antaa jäsenyysfunktion Z ja neljäs vaakarivi jäsenyysfunktion P vastaavat y-koordinaatit. Tämä matriisiesitys mahdollistaa sumean säätimen käsittelyn myös MATLAB- funktiona.

Olkoot tarkat mittausarvot eräällä hetkellä esimerkiksi: *e(k) = -0.3* ja *de(k) = 0.1*. Seuraavaksi lasketaan mittausarvojen jäsenyysasteet sumeissa joukoissa N, Z ja P, jolloin tuloksena saadaan mittausten sumeutetut arvot:
e = -0.3 = erosuureen täsmällinen arvo; *me = [0.30 0.70 0]* = erosuureen sumeutettu arvo *de = 0.1* = erosuureen muutoksen täsmällinen arvo on *mde = [0 0.80 0.20] = erosuureen* muutoksen sumeutettu arvo. Sumeutettu arvo on siis vektori, jonka komponenttien lukumäärä on sama kuin kyseisen suureen sumeiden joukkojen lukumäärä. [2]

Sääntökannan muodostaminen

Sumean säätimen toiminnalliset ehdot tallennetaan säätimen sääntökantaan. Asiantuntijan tietämykseen perustuvat säännöt luodaan yleensä muotoon, JOS (joukko ehtoja toteutuu), NIIN (joukko seurauksia voidaan päätellä). Jokaiseen sääntöön siis sisältyy edeltävä ehto-osa (antecedent condition) ja seurausosa.
Sumean säädön toiminta riippuu oleellisesti paitsi sääntökannasta myös prosessin mittaustietojen ja säätömuuttujien sumeutuksesta ja siihen liittyvistä sumeiden joukkojen jäsenyysfunktioista. Mitään yleispätevää ohjetta suunnittelun pohjaksi ei ole toistaiseksi voitu kehittää, vaan sumean säädön suunnittelussa on turvauduttava kokeiluun ja ennakolta tapahtuvaan simulointiin. [2]

Asiantuntijat, joilla on kyseisen prosessin ohjaamiseen tarvittava tietämys ja kokemus voivat laatia prosessin eri ohjaustilanteisiin liittyvät operointiohjeet, joiden perusteella sitten kirjoitetaan sumean säädön sääntökanta. Kokenut operaattori saattaa pystyä kirjoittamaan tietämyksensä myös suoraan JOS... NIIN...-sääntöjen muotoon. Tämän jälkeen sääntökantaa tarkennetaan kokeilujen ja testausten kautta.
Prosessin operaattorina toimiva ihminen toimii itse asiassa sumean säätöjärjestelmän tavoin. Hän toteuttaa, ehkä tiedostamattaan, JOS...NIIN...-tyyppisiä sääntöjä suorittaessaan ohjaus- ja säätötoimenpiteitä. [2]

Sääntökanta voidaan eräissä tapauksissa rakentaa myös prosessista tehdyn kuvauksen perusteella. Tämä suunnittelumenetelmä on mutkikkaampi ja työläämpi kuin muut menetelmät. Jos prosessin kuvaus on kattava, sen perusteella saatu sääntökanta on luotettava. On todennäköistä, että prosessien toimintakuvausten tekemiseen tullaan kehittämään formaalisia menetelmiä, joiden tuottaman tiedon pohjalta säätösuunnittelu voidaan toteuttaa luotettavasti niin, että säätimille saadaan parempi teoreettinen rakenne. [2]

Sumea säätö voi paitsi jäljitellä inhimillistä operaattoria ja myös pitää sisällään oppimismekanismin. Oppiva säädin kykenee muokkaamaan olemassa olevia sääntöjä sekä luomaan uusia sääntöjä. Se omaa hierarkkisen rakenteen, jossa on kaksi sääntökantaa. Ensimmäinen sääntökanta on yleissääntökanta, jonka perusteella säätö toimii. Toinen sääntökanta koostuu ns. metasäännöistä, jotka muuntelevat yleissääntökantaa sen mukaan miten kokonaistoiminnalle asetetut vaatimukset täyttyvät. Sääntökannoissa käytettävät säännöt voidaan jakaa kahteen ryhmään a) tilan arviointiin perustuvat säännöt ja b) tavoitteen arviointiin perustuvat säännöt. [2]

Tilan arviointiin perustuvat säännöt ovat muotoa: *JOS x on Ai ...ja y on Bi, niin z = fi (x, ..., y)*. Tämän tyypin säännöt arvioivat prosessin tilan hetkellä t ja määrittelevät sumean ohjauksen hetkellä t tulomuuttujien x ja y funktiona.
Tavoitteen arviointiin perustuvat säännöt arvioivat säädön tuloksena saatavan tilan (x, y) ja päättelevät sen avulla tarvittavan ohjauksen. Säännöt ovat tällöin muotoa: *JOS (u on C i implikoi (x on A i ja y on B i)), niin u on C.*

Sumeiden ohjausten käytännön sovellutuksissa tilan arviointiin perustuvat säännöt ovat yleisimmät. Monissa käytännön tapauksissa hyviä tuloksia on saavutettu käyttämällä P-, PI-, tai PD- tyyppisen säätimen sääntöjä. Sääntöjen optimaalisen lukumäärän selvittämiseksi ei ole olemassa mitään yleistä menetelmää. Sovellettaessa sumeaa säätöä johonkin määrättyyn prosessiin, kokeilut on syytä aloittaa varsin pienellä sääntökannalla. Tällöin yksittäisten sääntöjen vaikutus näkyy säätötuloksessa selvimmin. Vähitellen sääntökantaa voidaan lisätä tarkentamalla sääntöjä. Sääntökannan tulee olla johdonmukainen. Jokaisen säännön vaikutusta on tutkittava mahdollisimman perusteellisesti ja sääntöjen väliset ristiriidat on eliminoitava.[2]

Sumea päättely

Sumeassa päättelyssä sumeutetun mittauksen ja sääntökannan avulla päätellään tarvittava ohjaus käyttäen sumeita loogisia operaatioita kuten *and, or* ja *not* sekä *sumea implikaatio.* Sumeassa päättelyssä voidaan käyttää joko yleistettyä eteenpäin päättelyä (*Generalized Modus Ponens,* GMP) tai yleistettyä taaksepäin päättelyä (*Generalized Modus Tollens,* GMT). Säätösovelluksissa käytetään yleisimmin eteenpäin päättelyä. [2]

Sumeutus ja selkeytys

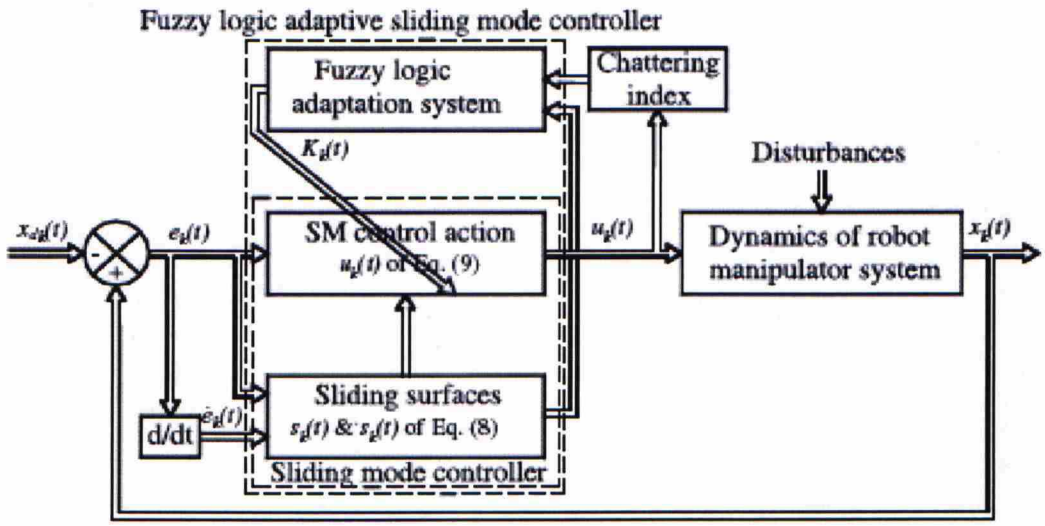
Sumeutus (Fuzzification) tarvitaan muuntamaan tulosuureiden xi tarkat numeeriset arvot sumean relaation muotoon. [2]

4.2.3. Robottisovelluksen parantaminen sumean logiikan laskennalla

Yleistä [17]

Sumeaa säätöä on käytetty usealla eri hierarkkisella tasolla robottien ohjauksessa kuten tehtävien suunnittelussa, järjestelmän monitoroinnissa, suodatuksessa ja ennalta prosessoinnissa sekä sisäisessä suorassa ohjauksessa. Monissa tosiaikaisissa eli real-time sovelluksissa vaaditaan vähintään 100 Hz:n kaistanleveyttä ja tämä on ollut suuri rajoite käytössä olevilla robottiohjaimilla.
Yleisesti on tiedossa se, että liukuvan eli jatkuvatoimisen laskennan ohjauksen algoritmeilla voidaan saada hyvä hetkellinen säätötarkkuus systeemille ja myös hyvä vastaavuus ohjauksen ulkoisille häiriöille (robustness). Robottien reaaliaikaisessa radan muodostamisessa (contour tracking) on ollut ongelmana käsivarren värähdysliike.

Tätä värähtelyn poistamista on pyritty parantamaan adaptiivisella säädöllä, jossa on hyödynnetty sumean logiikan laskenta-algoritmeja.



Kuva 4.11 Esimerkki sumean säädön lisäämisestä robotin ohjaimen yhteyteen matemaattisena mallina lohkokaavioesityksen muodossa. [17]

Tämä toimintamalli toteutettuna Fuzzy - logiikkateknologialla on kehitetty parantamaan robotin adaptiivista ohjausta. Robotin radan jäljityksen muodostaminen jatkuvatoimisella säätimellä lisättynä sumealla säädöllä parantaa itse säätimen tehoa, radan seurannan suorituskykyä, vaimentaa ohjaimen toiminnan värähtelyä sekä minimoi saavutettavan vaiheajan. Tärkeätä on huomioida se, että tällaisella järjestelmällä saavutetaan huomattavasti nopeampi radan muodostamistoiminta robotin ohjauksessa kuin ilman sitä.

Esimerkki sumean säädön käytöstä robottiohjauksessa [13]

Tässä esimerkkitutkimuksessa on esitelty kolmivaiheinen aktiivinen jäysteen poisto-strategia, joka perustuu Fuzzy-logiikkaan ja voimaohjaukseen. Järjestelmä perustuu automaattiseen pinnan seuraamiseen ennalta määrittelemättömän kappaleen pinnalla. Robotin liike ja voimaohjaus tapahtuu aktiivisesti havaitsemaan tarvittavan jäysteen poiston määrän ja paikoituksen epätarkkuuden optimaalisesta paikasta. Sumean säädön käyttöliittymällä saadaan yksinkertaistettua juuri jäysteen poiston oikeata määrää kussakin tilanteessa. Tällaisille sovelluksille olisi käyttöä muovi- ja metalliteollisuudessa valmistettaville tuotteille, joita valmistetaan valamalla, meistämällä tai poraamalla, jolloin vaaditaan pintojen viimeistelyä irrottamalla jäyste pois. Roboteilla suoritettaessa jäysteen poistoa ovat olemassa seuraavat vaikeudet:

- ✓ Usein kappaleet ovat monimutkaisia, jolloin robotin liikepaikoitusten opettaminen on työlästä niiden vaatiman äärimmäisen hyvän tarkkuuden vuoksi.
- ✓ Mahdolliset virheet/poikkeamat kappaleen sijainnissa, kappaleen mitta-poikkeamat sekä työkalun kuluminen aiheuttavat vielä lisävaikeuksia luotaessa robotin työstörataa.
- ✓ Etukäteen on mahdotonta myös ennustaa tarvittavan jäysteen poiston määrää.

Jäysteen poistomenetelmä
Poistettaessa robotilla jäystettä kappaleesta joudutaan tekemään seuraavat kolme vaihetta:

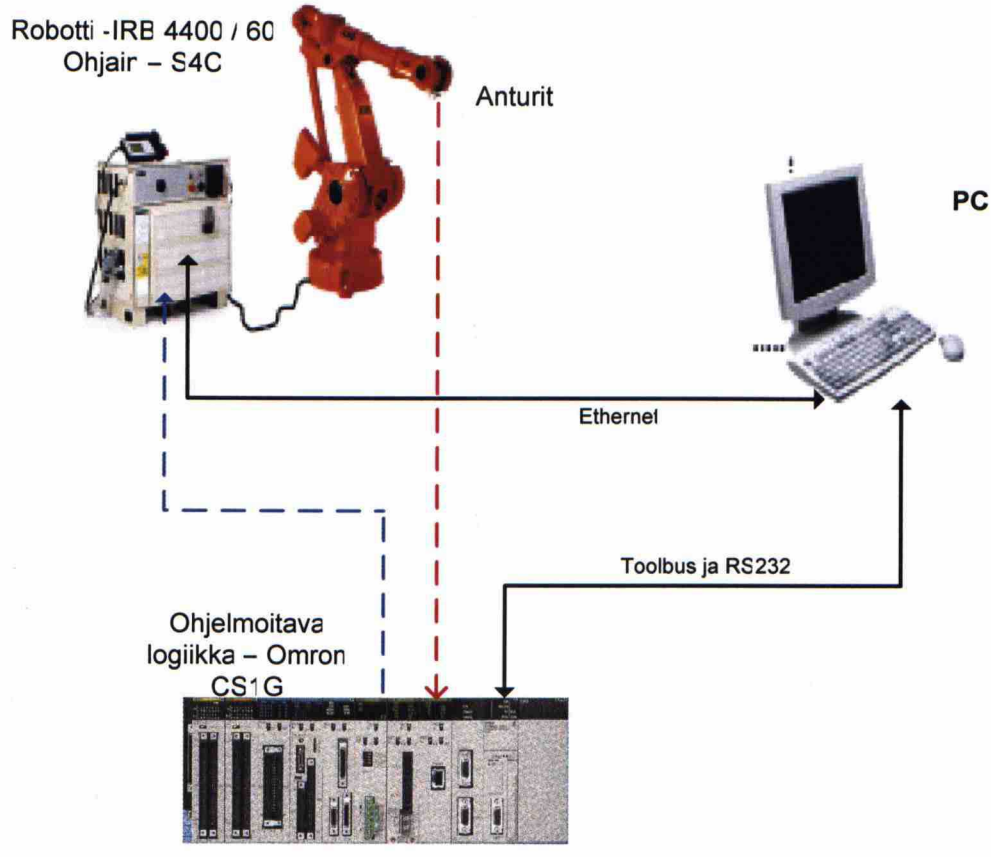
- ✓ haluttu robotin liikerata
- ✓ poistettavan jäysteen sijainti ja määrä on havaittava
- ✓ määriteltävä menetelmä kuinka jäyste poistetaan

Ensimmäisessä vaiheessa suoritetaan radan muodon seuranta valmiin mallikappaleen avulla. Toisessa vaiheessa suoritetaan sama tapahtuma, mutta sellaiselle kappaleelle, josta täytyy poistaa jäystettä ja tällöin muodostettu rata eroaa juuri jäysteen poiston kohdissa. Kolmas vaihe on luonnollisesti itse jäysteen poistotapahtuma näin ennalta muodostetun robottiradan mukaan. Tällöin mukana on myös voimaan perustuva ohjaus, mikä siis mahdollistaa mukautuvan eli adaptiivisen ohjauksen poistettavan jäysteen mukaisesti.
Tässä tutkimuksessa robotilla toteutettu jäysteen poisto perustui voima-anturilta tulevaan takaisin kytkentätietoon. Kolmessa vaiheessa toteutettu aktiivinen jäysteen poistomenetelmä ja sen perustella muodon seurantamenetelmä on myös siis kokeellisesti onnistuneesti tehty lisäämällä siihen Fuzzy -säätö mukaan, jolloin käyttäjän kokemusperäinen tieto esimerkiksi poistettavan jäysteen määrästä on helpompi määritellä.

5. TOTEUTETUN JÄRJESTELMÄN KUVAUS

5.1 JÄRJESTELMÄN YLEISKUVAUS

Toteutetussa robottijärjestelmässä käytettiin ABB IRB 4400 – robottia varustettuna S4C–robottiohjaimella. Nämä laitteet hankittiin Lahden ammattikorkeakoulun Tekniikan laitokselle loppuvuonna 1999. Robottiohjaimen osalta on tapahtunut tämän jälkeen kehitystä, sillä vuonna 2000 tuli markkinoille S4C+ -ohjain ja vuonna 2004 IRC5 – ohjain, mutta itse manipulaattori on pysynyt muuttumattomana.
Tutkimusprojektia varten hankittiin keväällä 2005 ohjelmoitava logiikka Omron CS1G ja siihen tarvittavat lisäkortit kuten analogiset AD/DA-muuntimet sekä Fuzzy–säädinkortti. Päätietokoneena käytettiin Intelin Pentium III 1 GHz:n prosessorilla varustettua PC:tä, jossa oli keskusmuistia 512 MB. Robotti ja PC kommunikoivat keskenään Ethernetin avulla. Ohjelmoitava logiikan Fuzzy–säädin liitettiin PC:hen RS232–yhteydellä ja muussa logiikan ja PC:n välisessä kommunikoinnissa käytettiin vähän nopeampaa toolbus–yhteyttä.

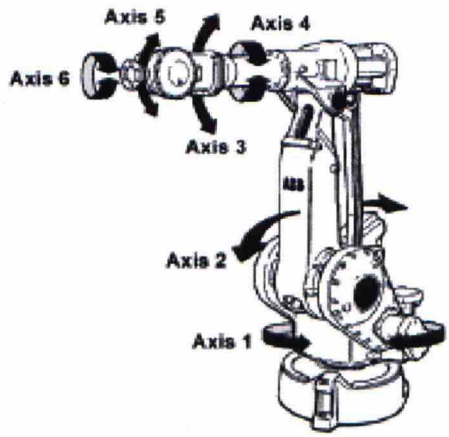


Kuva 5.1 Yleiskuva toteutetusta järjestelmästä ja sen laitteiden välisistä tiedonsiirtoyhteyksistä.

5. 2 ROBOTTI JA OHJAUSYKSIKKÖ

5.2.1 ABB IRB 4400/60

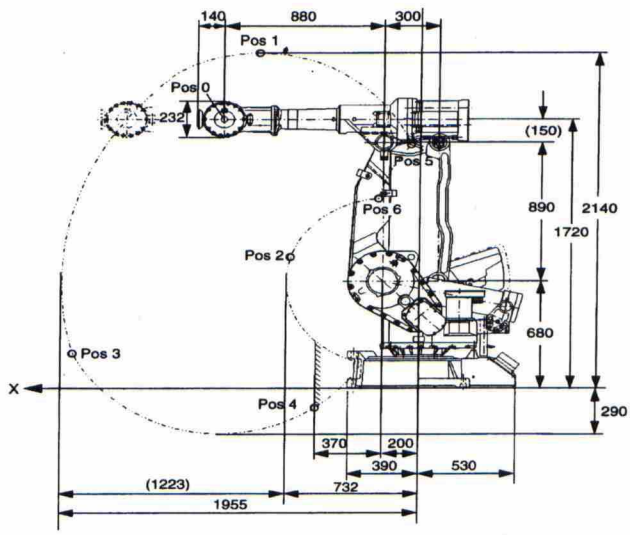
ABB IRB 4400 on kuuden akselin nivelvarsityyppiä oleva teollisuusrobotti, jota on paljon käytetty esimerkiksi autoteollisuuden hitsaustehtävissä, tuotteiden paikoitus- ja poimintatehtävissä sekä kokoonpanotehtävissä. IRB 4400 – robottia on saatavilla erilaisilla kuorman kantokyvyillä alkaen 10 kg:sta aina 60 kg:n kantokykyyn asti. Tässä tutkimusprojektissa oli käytössä 60 kg:n malli, jolla saavutetaan noin 2 metrin toimintasäde. Akseleiden toimilaitteina robotissa käytetään AC – servomoottoreita, joita on siis kuusi kappaletta. Paikoituspisteiden teoreettinen toistotarkkuus on 0.07 mm – 0.1 mm, mutta ohjelmoidun radan toistotarkkuus nopeudella 1 m/s on 0.25 mm – 0.4 mm. Liikenopeuden ISO – testi, jossa kaikki robotin akselit liikkuvat samanaikaisesti, antaa työkalupisteen maksimi nopeudeksi 2.2 m/s ja kiihtyvyydeksi 12 – 14 m/s^2. Robotin paino on 1000 kg.



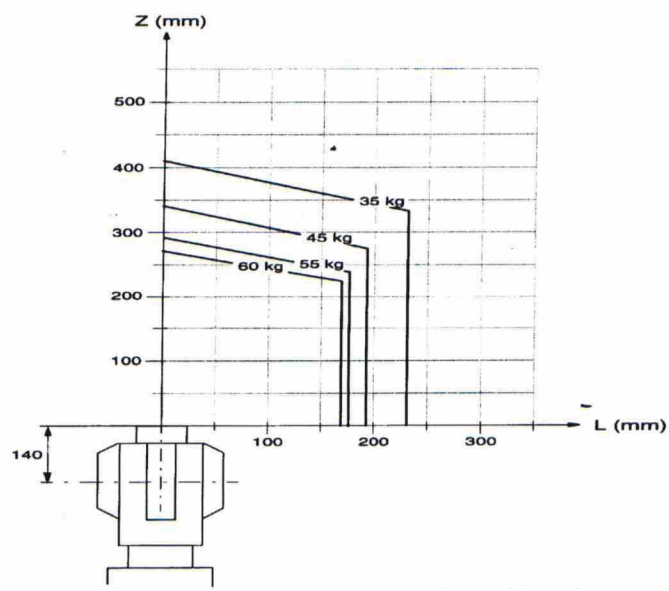
Akseleiden liikeradat:

1. akseli 330°
2. akseli 175 °
3. akseli 125 °
4. akseli 400 °
5. akseli 240 °
6. akseli 800 °

Kuva 5.2 IRB 4400 – robotin vapausasteet eli liikkuvat robottiakselit. [19]



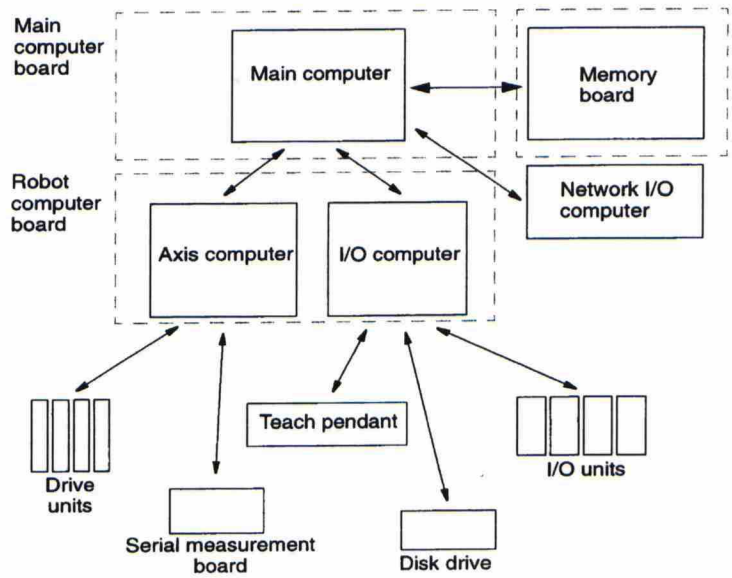
Kuva 5.3 IRB 4400 – robotin työalue. [19]



Kuva 5.4 IRB 4400 – robotin kantokyky luvatulla paikoitustarkkuudella, kun hyötykuorma sijaitsee tietyllä etäisyydellä robotin laipan keskipisteestä. [19]

5.2.2 S4C ohjain

Robottiohjaimen S4C:n tärkeimmät osat ovat pää- (main), akseliohjaus- (axis) ja I/O-tietokone. Kaikki nämä kolme erillistä tietokonetta vaaditaan, kun luodaan, ajetaan ja tallennetaan robottiohjelmia.



Kuva 5.5 Robottiohjaimen S4C lohkokaavio. [19]

Päätietokoneen (main computer) tärkein tehtävä on ohjata muiden laitteiden toimintaa. Robotin kuutta AC–servomoottoria ohjataan luonnollisesti akselitietokoneella (axis computer). Päätietokoneelta lähetään robotin paikoitusparametrit akselitietokoneelle ja sen hetkisen aseman akselitietokone saa sarjaliikenteen mittauskortilta (serial measurement board). Näitä tietoja se käyttää säätäessään parhaimmat algoritmit, jotta robotti voidaan liikuttaa vaadittuun asemaan annetulla nopeudella ja tietyllä kuormalla.
I/O–tietokoneen tehtävänä järjestää kommunikointi ulkoisiin laitteisiin, joita ovat esimerkiksi teach pendant–käsiohjelmointiyksikkö, levykeasema ja fyysiset input- ja outputkortit.
Muistiyksikkö (memory board) sisältää 16 MB RAM–muistia ja päätietokoneen muistikapasiteetti on 8 MB. Digitaalisessa I/O–kortissa on 16 tuloa ja 16 lähtöä ja lisäoptiona hankitussa analogisessa I/O–kortissa on 4 analogista tuloa ja 4 analogista lähtöä. Analogisiin tuloihin (-10 V/+10 V) on liitetty nyt ohjelmoitavan logiikan analogiset lähdöt (sumeat lähdöt), kun taas logiikan analogisiin tuloihin (sumeat tulot) on liitetty käytetyt analogiset anturit. Robotin analogisista lähdöistä kolmella voidaan käyttää -10 V/+10 V – signaalitasoa ja yhdellä 4 – 20 mA virtaviestiä.
Analogiset tulot ovat 14 bittisiä, joten resoluutio on 16384 jakoinen eli yhden bitin muutosta vastaa 0,61 mV jännitteen muutos (10 000 mV / 16384). Päivitysväli (time intervals) on laitteiston osalta 2 ms, mutta robotin ohjausjärjestelmä rajoittaa käytännössä arvon 4 millisekuntiin, jolloin päivitystaajuus on 4 kHz. Ruotsalaisten suorittamissa koeajoissa [10] ilmeni, että pahimmassa eli hitaimmassa tapauksessa viive voi olla jopa 120 ms, jolloin anturitiedon päivitystaajuus on vain noin 10 Hz.

Ulkoiseen sarjaliikennemuotoiseen liikennöintiin ohjaimessa on kaksi sarjaporttia, joihin voidaan liittää tulostin, tietokone, ohjelmoitava logiikka, konenäköjärjestelmä tai jokin muu sarjaliikennettä käyttävä laite. Molempien porttien nopeusalueet ovat välillä

300 - 19 200 baudia. Toinen käyttää RS 232–protokollaa tukien RTS-CTS – ohjausta lisättynä XON/XOFF – ohjauksella sekä toinen kaksi suuntaista (full duplex) RS 422 – protokollaa. Tässä tutkimusprojektissa ulkoinen tietokone oli liitetty robottiohjaimeen kuitenkin Ethernetin avulla. Robotin Ethernet–kortti tukee IEEE 802.3:n 10BASE-T yhteyttä, joten maksiminopeus on 10 Mbit/s. Maksimietäisyys kytkimestä, keskittimestä tai tietokoneesta voi olla 100 metriä.

5.2.3 Robotin ohjelmisto

ABB Oy toimittaa valmistamansa robotit ohjelmistolla RobotWare, joka kostuu ABB:n Automation Technology Product Oy:n kehittämistä kolmesta erillisestä ohjelmatuotteesta.

RobotWare – ohjelmistoperhe koostuu:

- BaseWare OS
- BaseWare Options
- ProcessWare

BaseWare OS on robotin käyttöjärjestelmä eli muodostaa siis RobotWare–ohjelmisto - perheen ytimen (kernelin). Tämä on luonnollisesti kaikissa toimitetuissa roboteissa, koska se on kaiken robotin toiminnan perustana. Tutkimusprojektin robotissa on käytössä BaseWare OS versio 3.1.
BaseWare Options sisältää useita erillisiä toiminnallisia ohjelmistopaketteja kuten RAP – kommunikoinnin, Ethernet – palvelut jne..
ProcessWare tarjoaa tuotteita, jotka ovat räätälöity esimerkiksi hitsaussovelluksiin tai maalaustehtäviin. ProcessWare-ohjelmistolisäysten avulla näissä prosesseissa käytettävissä roboteissa ohjelmointi yksinkertaistuu huomattavasti.
Tutkimusprojektin robotissa on tällä hetkellä kaikki mahdolliset BaseWare Options – ohjelmapäivitykset, jotka voidaan asentaa BaseWare OS 3.1 – käyttöjärjestelmään. Seuraavaksi annetaan lyhyt kuvaus niiden ominaisuuksista.

Advanced Motion
Tämä lisäohjelmisto hankittiin kesällä 2005 tutkimuksessa käytettävään robottiin. Ohjelmisto oli välttämätön, jotta voidaan toteuttaa ns. 'contour tracking' -toiminta eli muodon seuranta yhden tai useamman analogisen anturin avulla. Ulkoisen adaptiivisen ohjauksen kehittämisen kannalta tämä on luonnollisesti tärkein toimintomuoto. Käytännön kokeilut ja ohjelmien rakenneideat esitellään luvussa kuusi, jossa käsitellään toteutetut ratkaisut ja koeajot.
Advanced Motion – paketti tarjoaa myös kolme muuta lisätoimintoa 1) yksittäisen akselin työalueen resetointi 2) itsenäisten eli toisista akseleista riippumattomien liikkeiden mahdollistaminen 3) rinnakkaisten liikkeiden suorittaminen samanaikaisesti robotilla ja ulkoisella akselilla.

Multitasking
Tämä lisäohjelmisto hankittiin joulukuussa 2005 tutkimuksessa käytettävään robottiin, koska muodon seurannan toimintaa haluttiin vielä yrittää parantaa. Multitasking –

ohjelmisto mahdollistaa maksimissaan kymmenen ohjelman (taskin) rinnakkaisen suorittamisen samanaikaisesti varsinaisen robotin pääohjelman kanssa, mutta luonnollisesti ainoastaan pääohjelma voi ohjata robottia liikkumaan paikasta toiseen. Tyypillinen käyttösovellutus on ohjaimen sarjaportin lukeminen samanaikaisesti robotin tehdessä liikekäskyjä.

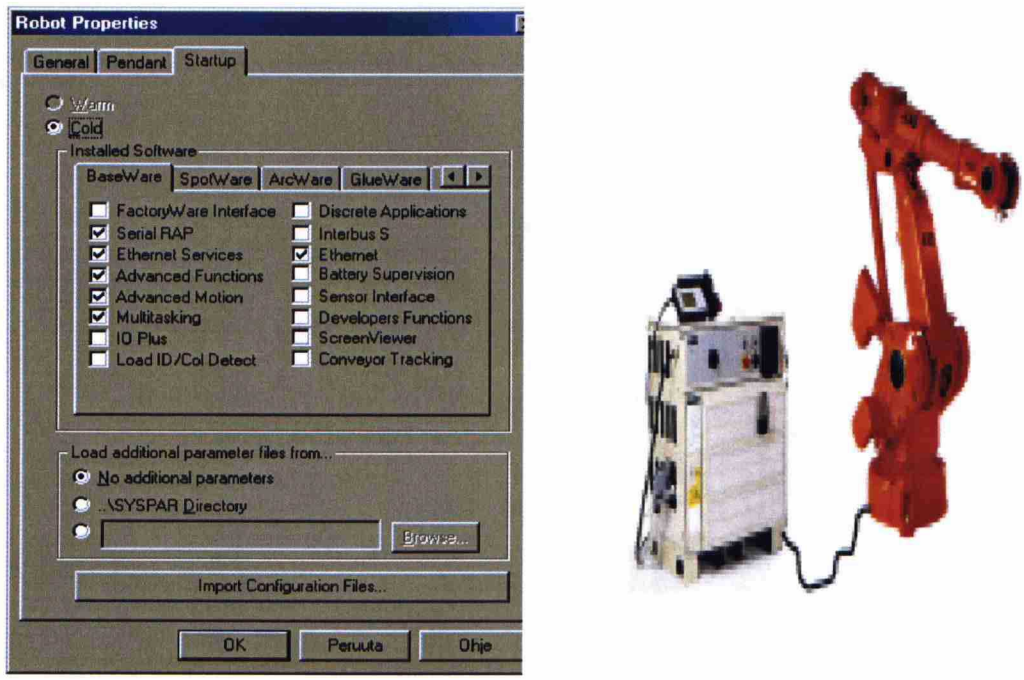
Advanced Functions
Tämä ohjelmistopaketti on lähes välttämätön päivityspaketti robotin käyttöjärjestelmään, joten se on mukana lähes poikkeuksetta robottitoimituksessa.
Se mahdollistaa seuraavat toiminnot:

- tiedon siirto sarjaporttia tai tiedostoja käyttäen
- fyysisen lähdön asettaminen päälle tietyssä robotin paikoituspisteessä
- fyysisen tulon arvon tarkastus tietyssä robotin paikoituspisteessä
- aliohjelman suoritus tietyssä robotin paikoituspisteessä
- kiellettyjen tai / ja sallittujen alueiden määrittely robotin ohjelmassa
- automaattinen lähdön päälle asetus robotin ollessa tietyllä ennalta määritellyllä alueella
- robotin liikkeen mahdollistaminen virhetilanteissa, kun virheenkäsittelyohjelma tai keskeytysohjelma on käynnissä
- loogisten ehtojen yhdistäminen yhdeksi loogiseksi rakenteeksi
- keskeytysaliohjelman kutsuminen myös analogisten tietojen perusteella

RAP -Communication
Tämä ohjelmisto mahdollistaa robottiohjaimen ja tietokoneen välisen kommunikoinnin Robot Application Protocol (RAP) -protokollaa käyttäen. Sitä voidaan käyttää sekä sarjaliikenne- ja Ethernet -yhteyksissä. Rap-kommunikointi mahdollistaa robotin ohjelman hallinnan etätietokoneelta. Tällöin robotin ohjelmalle voidaan suorittaa seuraavia etähallintatoimintoja:

- voidaan käynnistää tai pysäyttää
- ohjelmien siirto
- systeemiparametrien muuttaminen tai siirtäminen
- robotin tilan lukeminen
- tulojen tilan lukeminen tai lähtöjen lukeminen sekä asettaminen
- virheilmoitusten ja – logien lukeminen
- robotin tilan muuttaminen esimerkiksi opetustilasta tuotantotilaan

Ethernet Services
Se mahdollistaa kommunikoinnin etätietokoneen ja robottiohjaimen välillä joko ftp – protokollaa käyttäen tai Unix-perusteista NFS- tiedonsiirtoa hyväksi käyttäen. Tutkimusprojektissa robottiohjelmat tallennettiin myös etätietokoneen jaetulle levyasemalle, jolloin kyseisessä tietokoneessa vaadittiin myös NFS-server -ohjelmisto, jolloin robottiohjain pystyi automaattisesti ottamaan yhteyden jaetulle levyasemalle. Tätä toimintaa käytettiin myös robotin suorittamien taulukkoon tallennettujen anturimittaustietojen siirtämiseen etätietokoneessa olevaan Microsoft Excel – taulukkolaskentaohjelmaan.



Kuva 5.6 Virtuaalirobottiohjaimen (QuickTeach PC-ohjelmisto) S4C BaseWare Options –pakettien käyttöönotto ja kylmä- eli cold – boottauksen suorittaminen.

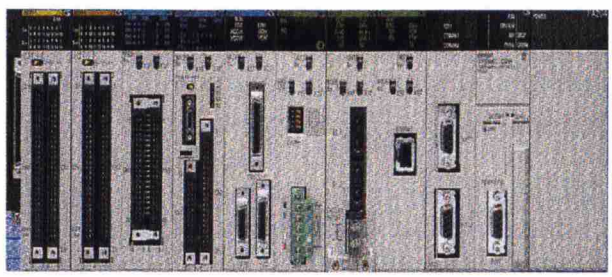
5.3 OHJELMOITAVA LOGIIKKA JA SUMEAN SÄÄDÖN YKSIKKÖ

5.3.1 Omron SYSMAC CS1G-H ohjelmoitava logiikka

Tutkimusprojektiin valittiin suuren japanilaisen automaatiokomponentteja valmistavan OMRON Oy:n SYSMAC – CS1 sarjan ohjelmoitava logiikka. Tähän kooltaan keski-suuren logiikkasarjan laitteisiin löytyy lähes kaikki mahdolliset lisäkortit kuten tässä projektissa tarvittava sumean säädön yksikkö. Erilaisia keskusyksiköitä on valittavissa yhdeksän kappaletta, joiden suoritusarvot vaihtelevat melko paljonkin toisistaan. Projektin logiikan keskusyksikkö CPU42H on sarjan edullisin. Siinä on ohjelmamuistia 10 000 riviä ja datamuistia 64 000 sanaa (16 bittinen sana) eli 128 kB. Sen prosessorin kellotaajuudeksi muodostuu 250 MHz, koska yhden käskyn suoritusaika on 0.04 mikrosekuntia eli 40 nanosekuntia.

CS1-logiikka edustaa SYSMAC -logiikoiden terävintä huippua. Fyysinen koko on sama perinteikkään C200H -sarjan kanssa, mutta suorituskyky on kasvanut monta kerta-luokkaa. CS1 on oikea valinta nopeutta, monipuolista käskykantaa, laajoja muistialueita, hyviä tiedostokäsittelyominaisuuksia tai tehokasta kommunikointia vaativiin sovelluksiin. Monipuoliset I/O-kortit laajentavat sovellusmahdollisuuksia. Väylätekniikoita löytyy useita, tuttujen Ethernet ja Controller Link-väylien lisäksi, voidaan käyttää myös DeviceNet-väylää tai Profibus-kenttäväylää. CS1-logiikkaan

voidaan kommunikoida jopa kolmen eri verkkotason läpi. Ethernet -väylän kautta voidaan siirtää tiedostoja tietokoneen ja CS1:n muistikortin välillä käyttämällä FTP-protokollaa. Muistikortille voidaan helposti kerätä mittaustietoa kentältä tai lukea kortilta logiikalle valmistukseen tarvittavia tietoja. Myös sähköpostia voidaan lähettää Ethernet -yksikön kautta esimerkiksi päivystäjälle prosessin häiriötilanteessa.



Ominaisuuksia (maksimiarvoja):
-maks. 5120 dig. I/O
-maks. 640 analogi-I/O
-ohjelmamuisti 250 K steps (1MB)
-datamuisti 448 Kwords
-CompactFlash-muistikortti
-e-mail (SMTP), ohjelmointi usean verkkotason läpi

Kuva 5.7 Omron SYSMAC – CS1 ohjelmoitava logiikka.

Tutkimusprojektin ohjelmoitava logiikka CS1G-H (CPU42H) varustettiin RS422 – liikennöintiportilla PC-yhteyttä varten, analogisella AD002-tuloyksiköllä robotissa olevia antureita varten, analogisella DA002-lähtöyksiköllä, sumean säädön C200H-FZ001 -yksiköllä sekä digitaalisilla tulo- ja lähtöyksiköillä.

Analogiseen AD002 -tuloyksikköön on mahdollista liittää kahdeksan analogista anturia, joiden viestitasot voivat olla 4 – 20 mA virtaviestejä tai -10 V/+10 V jänniteviestejä. Tutkimuksessa käytetyt laseranturit toimivat alueella 0 – 10 V ja voima – anturit 4 – 20 mA. Kortin ns. machine-numero asetettiin nollaksi, jonka jälkeen tehtiin alla olevat asetukset ja johdotukset.

Logiikan CIO (Common I/O alue) -alue määräytyy machine-numeron **n** perusteella CIO 20**n**0:

- CIO 2001 (ensimmäinen analogiatulo) → voima – anturi Y (4-20mA)
- CIO 2002 (toinen analogiatulo) → voima – anturi X (4-20mA)
- CIO 2003 (kolmas analogiatulo) → voima – anturi Z (4-20mA)
- CIO 2004 (neljäs analogiatulo) → laserZ (0 – 10 V)
- CIO 2005 (viides analogiatulo) → laserY (0 – 10 V)

Tarvittavat asetukset vastaavasti D20**n**00 kanavaan eli sanaan:

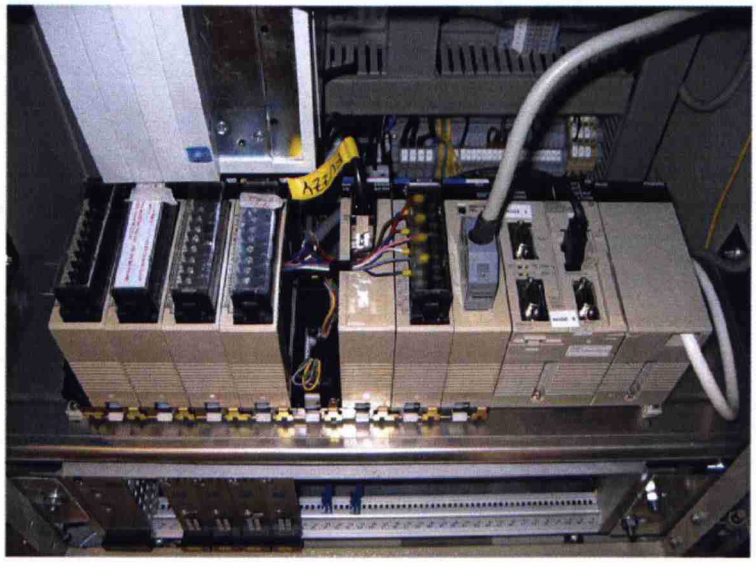
- käytössä viisi analogiatuloa ja tulos BCD – lukuna → D20000 sisällöksi annettava hexaluku 01E0
- kolme käyttämätöntä analogiatuloa, kolme virtaviestillä ja kaksi jänniteviestillä toimivaa → D20001 sisällöksi annettava hexaluku FD6A

Analogisessa DA002-lähtöyksikkössä on mahdollista käyttää neljää lähtöä, joiden viestitasot voivat olla 4 – 20 mA virtalähtöjä tai -10 V/+10 V jännitelähtöjä. Tutkimuksessa käytetyt lähdöt kytkettiin robotin analogiatulokortille ja ne toimivat alueella 0 – 10 V.
Kortin ns. machine-numero asetettiin ykköseksi, jonka jälkeen tehtiin seuraavalla sivulla olevat asetukset ja johdotukset.

Logiikan CIO (Common I/O alue)-alue määräytyy machine-numeron **n** perusteella CIO 20**n**0:

- CIO 2010 (analogialähtö 1)→ robotin X-akselin suuntainen voima tai liikekorjaus
- CIO 2011 (analogialähtö 2) → robotin Y-akselin suuntainen liike tai mitattu Z-koordinaattiarvo
- CIO 2012 (analogialähtö 3) → robotin Z-akselin suuntainen voima tai liikekorjaus
- CIO 2013 (analogialähtö 4) → mitattu Y-koordinaattiarvo

Anturien käytöstä riippuen kyseiset suuntien numeroarvot määräytyvät sumean ohjauksen tuloksina. Ohjelmoitavan logiikan ohjelman ja käyttöliittymän valinnan mukaisesti valitaan joko voima-anturien käyttö tai laserantureiden käyttö.



Kuva 5.8 Kuva tutkimusprojektin ohjelmoitavan logiikan kytkennöistä.

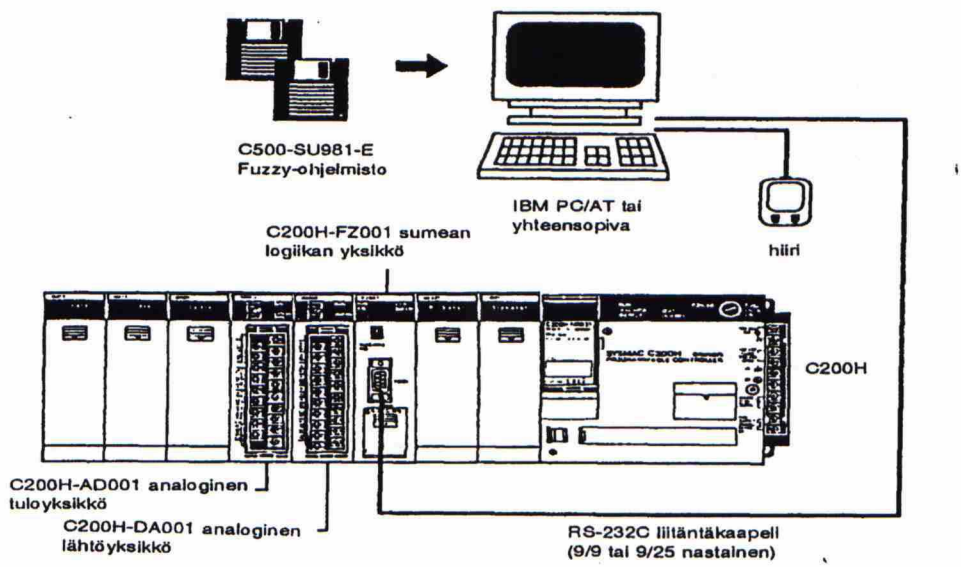
5.3.2 Omron SYSMAC C200H-FZ001 sumean logiikan yksikkö

C200H-FZ001 sumean logiikan yksikön avulla voidaan ohjelmoitavassa logiikassa käyttää nopeaa ja edistyksellistä sumean logiikan teknologiaa. Joidenkin monimutkaisten järjestelmien säätö, johon ennen tarvittiin paljon ohjelmarivejä, voidaan nyt suorittaa sumean logiikan yksikön avulla.

C200H-FZ001 yksikössä on mahdollista käyttää kahdeksaa tuloa ja neljää lähtöä. Jokaisessa säännössä voi olla kahdeksan ehto- ja kaksi päättelyosaa, jotka mahdollistavat sen käytön monissa sovelluksissa. C200H-FZ001 sumean logiikan yksikköä käytetään kuten muitakin I/O-yksiköitä. Sumean logiikan tukiohjelmistoa (FSS) käytetään tietämyskannan luontiin, sen lataamiseen logiikkaan sekä yksikön toimintojen seurantaan.

Kortin ns. machine-numero asetettiin kakkoseksi, jonka jälkeen tehtiin alla olevat asetukset.

- Kanavaan 2020(20n0) ilmoitetaan analogiatulojen määrä eli 5-7 kpl.
- Kanavaan 2021 ilmoitetaan sen sanan osoite, joka sisältää ensimmäisen analogiatulon datan (datamuistialue ja ensimmäinen kanava on 201).
- Kanavaan 2022 ilmoitetaan analogialähtöjen määrä eli 4 kpl.
- Kanavaan 2023 ilmoitetaan sen sanan osoite, joka sisältää ensimmäisen analogialähdön datan (datamuistialue ja ensimmäinen kanava on 110).
- Aloitetaan tai lopetetaan sumeaohjaus ohjaamalla bittiä 2020.15.



Kuva 5.9 Sumean säädön tarvitsemat laitteet [27]

5.4 ANTURIT

5.4.1 Laseranturit

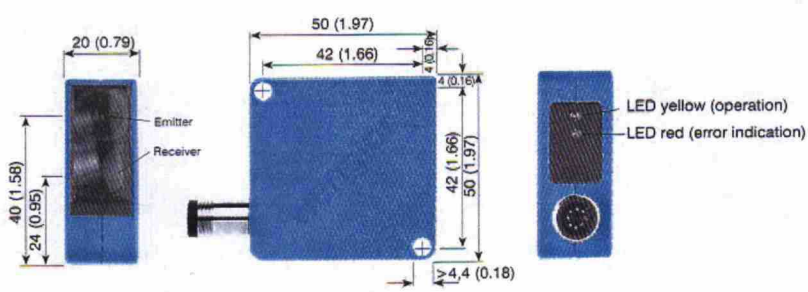
Laserantureina tutkimusprojektissa käytettiin Wenglorin valmistamia analogisia antureita, joita on kaksi kappaletta ja molempien analoginen signaalitaso on 0 – 10 V. Molemmat anturit sisältävät myös vahvistimen integroituna itse anturiosaan ja laserluokitus on kaksi. Toimintaperiaate perustuu kolmiomittausperiaatteeseen, jossa valolähde lähettää pulssittamalla valopisteen mitattavaan kohteeseen. Pulssittamalla estetään muun taustavalon vaikutus mittaamiseen. Syntyvän pisteen tai alueen kuva projisoidaan tietyssä kulmassa olevan optiikan kautta mittauselementille ja kohteen etäisyyden vaihdellessa liikkuu pisteen projektio valoherkkää elementtiä pitkin.

Wenglor YP11MGV-P24 ominaisuudet (Y-akselin suuntainen):

- toiminta-alue: 50...100 mm
- lineaarisuus: 1 %
- mittausetäisyys: 75 mm
- mittausalue: 50 mm (+/-25 mm)
- laser suojausluokka: 2 (EN 60825 – 1)
- valopisteen halkaisija: 1 mm
- resoluutio: 0.1 mm eli 100 µm
- vasteaika: 1 ms
- pulssitustaajuus: 1 kHz.

Wenglor YP 06MGVL80 ominaisuudet (Z-akselin suuntainen):

- toiminta-alue: 40...60 mm
- lineaarisuus: 0.5 %
- mittausetäisyys: 50 mm
- mittausalue: 20 mm (+/-10 mm)
- laser suojausluokka: 2 (EN 60825 – 1)
- valopisteen halkaisija: 0.5 mm
- resoluutio: 0.005 mm eli 5 µm
- vasteaika: 5 ms
- pulssitustaajuus: 100 Hz.



Kuva 5.10 Laserantureiden mittatiedot. (Anturiesite, Sensorola Oy)

Kytkettäessä laseranturit suoraan robotin analogiatuloihin joudutaan tekemään seuraavat parametriasetukset robottiin.

Y-akselin suuntainen laser:

- fyysinen minimiarvo: 0 V (etäisyys yli 100 mm)
- fyysinen maksimiarvo: 10 V (etäisyys 50 mm tai alle)
- looginen minimiarvo: 0
- looginen maksimiarvo 500

Mittausalueen ollessa 50 mm (50 mm...100 mm) ja resoluution ollessa 0.1 mm→ yhden yksikön muutos vastaa todellisuudessa 0.1 mm:n muutosta.

Z-akselin suuntainen laser:

- fyysinen minimiarvo: 0 V (etäisyys yli 60 mm)
- fyysinen maksimiarvo: 10 V (etäisyys 40 mm tai alle)
- looginen minimiarvo: 0
- looginen maksimiarvo 4000

Mittausalueen ollessa 20 mm (40 mm...60 mm) ja resoluution ollessa 0.005mm→ yhden yksikön muutos vastaa todellisuudessa 0.005 mm:n muutosta.

Kytkettäessä laseranturit ohjelmoitavan logiikan analogiatuloihin saadaan seuraavat resoluutiot yhden yksikön muutokselle (logiikan AD-muunnin on 12 bittinen→4096):

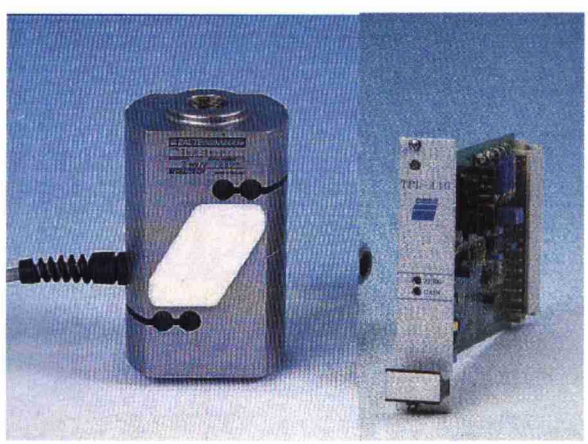
Y-akselin suuntainen laser:

- 50 mm / 4096 → 0.0122 mm, anturi kykenee kuitenkin vain 0.1 mm resoluutioon, jolloin anturin vastaava 0.1 mm:n muutos aiheuttaa 8 yksikön muutoksen logiikan AD-kortilla.

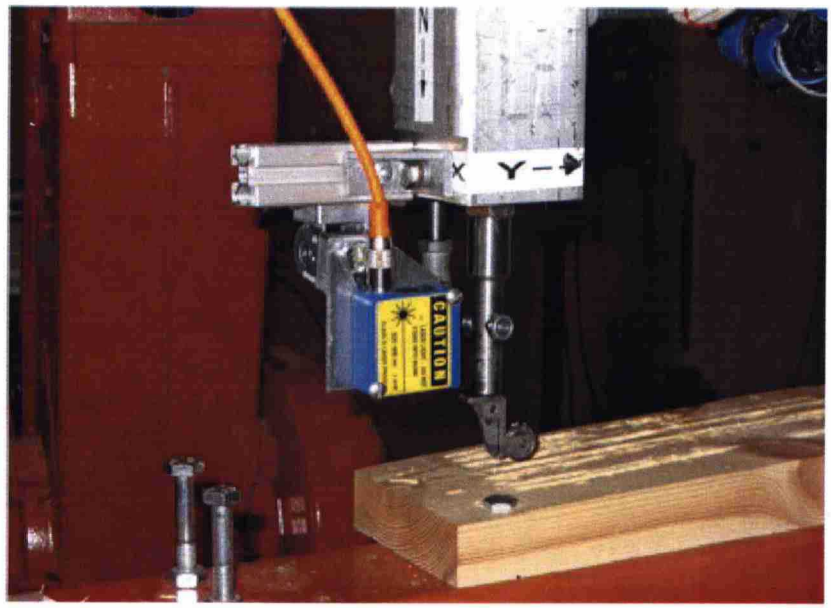
Z-akselin suuntainen laser:
20 mm / 4096 → 0.00489 mm, anturi kykenee kuitenkin nyt 0.005 mm resoluutioon, jolloin anturin vastaava 0.005 mm:n muutos vastaa lähes täysin yhden yksikön muutosta (0.00489→0.005) logiikan AD–kortilla.

5.4.2 Voima-anturit

Voiman mittausta varten toteutettiin oma voima-anturisovitin, joka kykenee mittaamaan voiman x-, y- ja z-suunnissa, mutta kiertoliikkeiden dx, dy ja dz mittausmahdollisuus puuttuu. Veto/puristusantureina käytettiin lahtelaisen Raute Oy:n valmistamia TB3 punnitusantureita, joita tarvittiin siis kolme kappaletta, joiden 300 kg:n maksimikuormitus oli ehkä vähän liian suuri. Käytetyt punnituslähettimet olivat tyyppiä TPL-110, joiden analogialähdöistä käytettiin 4-20 mA virtaviestiä.

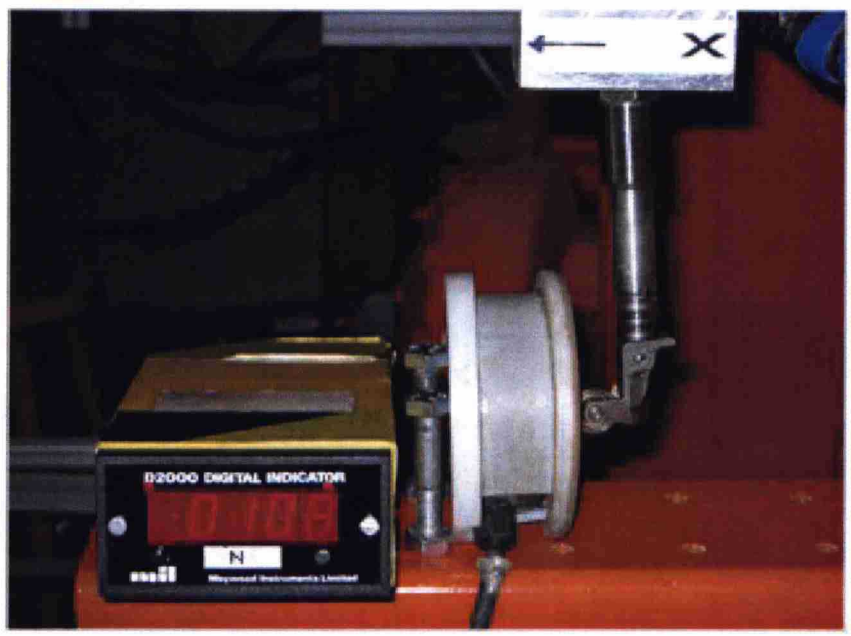


Kuva 5.11 Vasemmalla punnitusanturi TB3 ja oikealla siihen tarvittava TPL-110 vahvistinkortti.



Kuva 5.12 Robotin työkalulaippaan sijoitettu alumiininen omavalmiste, jossa on siis kolme punnitusanturia.

Voima-anturisovittimen kalibrointi suoritettiin kaikissa kolmessa suunnassa näyttämään vastaavia oikeita robotin käsivarteen kohdistuvia kuormitusarvoja.



Kuva 5.13 Voiman kalibrointi robotin x – liikesuunnassa.

Valmiita robotteihin tarkoitettuja työkalulaipan sovittimia voiman ja vääntömomentin mittausta varten on myös markkinoilla. Esimerkiksi Ruotsissa Västeråsin ABB Oy:n robottitehtaalla suoritettiin kesällä 2005 koeajoja erilaisista käyttösovellus-mahdollisuuksista ATI Industrial Automation Oy:n valmistamilla voima/vääntö-momenttilaipoilla.

Alla olevissa taulukoissa on ATI:n valmistamien voima -anturisovittimien tyypit:

Transducer Body	Axes	English	Metric
Nano17	Fxy Fz Txy Tz	± 79 lbs ± 180 lbs ± 23 in-lbs ± 28 in-lbs	± 350 N ± 800 N ± 2.6 N-m ± 3.1 N-m
Nano25	Fxy Fz Txy Tz	± 520 lbs ± 1400 lbs ± 310 in-lbs ± 560 in-lbs	± 2325 N ± 6250 N ± 34 N-m ± 62 N-m
Nano43	Fxy Fz Txy Tz	± 68 lbs ± 89 lbs ± 30 in-lbs ± 48 in-lbs	± 300 N ± 400 N ± 3.4 N-m ± 5.4 N-m
Mini40	Fxy Fz Txy Tz	± 200 lbs ± 610 lbs ± 190 in-lbs ± 190 in-lbs	± 870 N ± 2700 N ± 21 N-m ± 21 N-m
Mini 45	Fxy Fz Txy Tz	± 1200 lbs ± 2300 lbs ± 950 in-lbs ±1200 in-lbs	± 5100 N ± 10000 N ± 110 N-m ± 140 N-m
Gamma	Fxy Fz Txy Tz	± 270 lbs ± 910 lbs ± 690 in-lbs ± 730 in-lbs	± 1200 N ± 4100 N ± 79 N-m ± 82 N-m
Delta	Fxy Fz Txy Tz	± 770 lbs ± 2600 lbs ± 2000 in-lbs ± 3700 in-lbs	± 3400 N ± 12000 N ± 220 N-m ± 420 N-m
Theta	Fxy Fz Txy Tz	± 5700 lbs ± 14000 lbs ± 22000 in-lbs ± 24700 in-lbs	± 25000 N ± 61000 N ± 2500 N-m ± 2700 N-m
Omega160	Fxy Fz Txy Tz	± 4000 lbs ±11000 lbs ±15000 in-lbs ±17000 in-lbs	±18000 N ± 48000 N ±1700 N-m ±1900 N-m
Omega190	Fxy Fz Txy Tz	± 8000 lbs ± 25000 lbs ± 49000 in-lbs ± 72000 in-lbs	± 36000 N ±11000 N ± 5500 N-m ± 8100 N-m

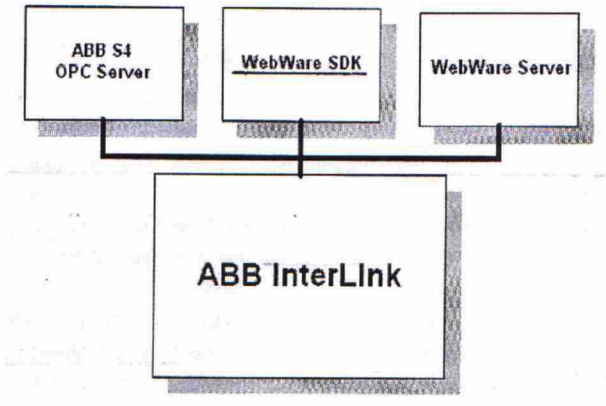
Taulukko 1. ATI:n valikoimat robotteihin sijoitettavista voima-anturisovittamista.(http://www.ati-ia.com/products/ft/ft_models.aspx)

Kuva 5.14 ABB:n robottitehtaalla suoritettu koeajokuva voiman ja vääntömomentin mittauksen käyttömahdollisuudesta auton automaattivaihteiston kokoonpanotyössä.

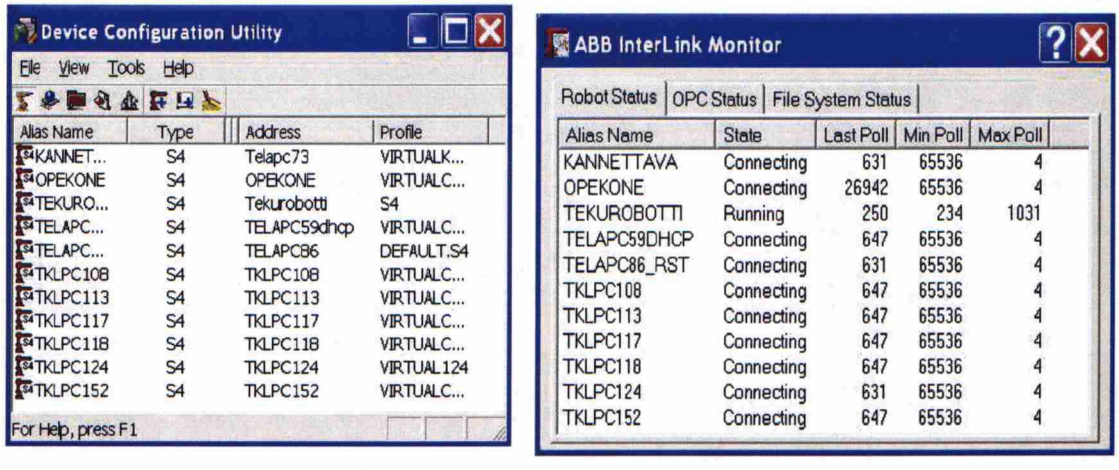
5.5 PC – OHJELMISTOT

5.5.1 ABB WebWare SDK 4.6 ja InterLink – ohjelmistot

PC-koneisiin kehitetyn käyttöliittymän ja robotin välisen kommunikoinnin toteuttamiseen käytettiin ABB WebWare SDK 4.6 -kehitystyökalua, jota voidaan hyödyntää Visual Basic 6 tai Visual Basic.Net ohjelmointiympäristöjen kanssa. InterLink-ohjelmisto tarjoaa OPC (OLE for Process Control) rajapinnan PC:n ja robotin välille käytettäessä esimerkiksi Ethernet-yhteyttä.
OPC on ohjelmistostandardi, jolla on helppo liittyä eri valmistajien automaatio-ratkaisuihin. OPC käyttää Microsoftin DCOM teknologiaa ja nykyään myös Linux/Unix ratkaisuja. OPC-Foundation järjestöön (http://www.opcfoundation.org/) kuuluu runsaasti jäseniä, jotka toimittavat automaatioon laitteistoja ja ohjelmistoja ja näin saadaan joustava tuki suurelle sovellusjoukolle. Lisäksi OPC käyttää verkkoa tehokkaasti ja sopii myös laajoihin sovelluksiin.



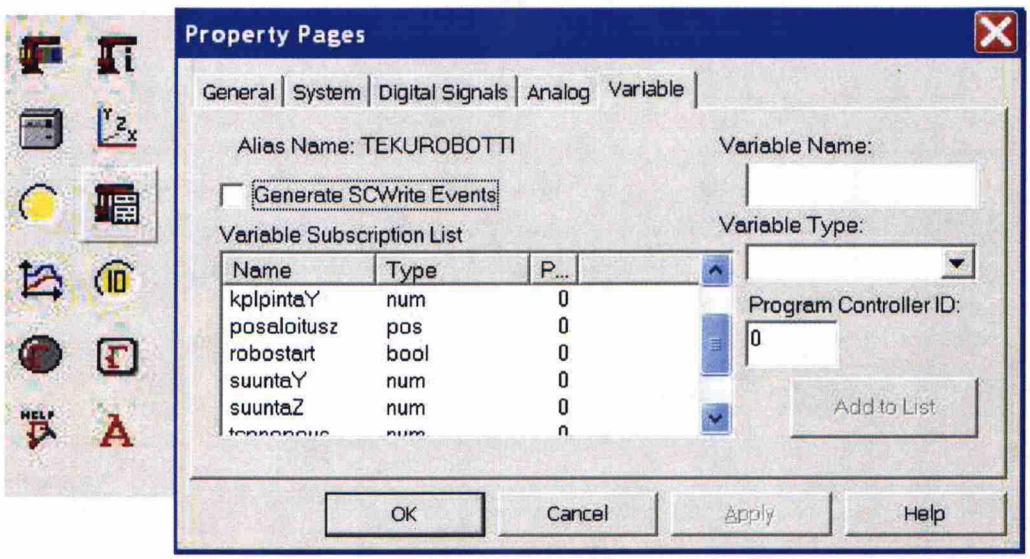
Kuva 5.15 Lohkokaavio ABB ohjelmistoista, joita voidaan käyttää PC koneiden ja robottien väliseen kommunikointiin.



Kuva 5.16 InterLink-ohjelmiston konfigurointi- ja monitorointi – ikkunat

WebWare SDK on erittäin monikäyttöinen kehitystyökalu, joka sisältää erilaisia metodeja, tapahtumia sekä objekteja, joita voidaan lisätä Visual Basicin työkaluihin. Ohjelmistopaketin manuaalissa on lähes 1000 sivua.

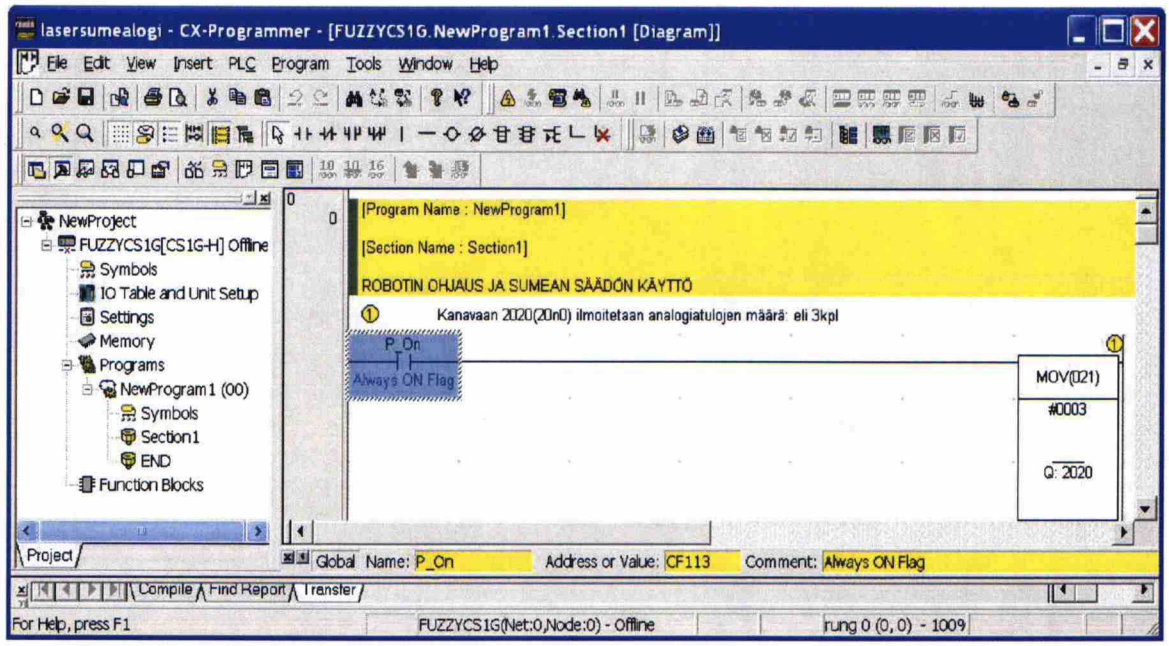
Kehitystyökalun yksi konekohtainen lisenssi maksaa noin 5000 €, mutta tähän tutkimusprojektiin sen hinnaksi muodostui 2000 €, koska Suomen ABB-robottiosasto tiedusteli suoraan Ruotsista erikoisalennusta. Huomioitavaa on kuitenkin se, että asennettaessa valmis käyttöliittymäsovellus johonkin toiseen PC-koneeseen tarvitsee suorittaa vain InterLink-ohjelmiston asennus ja tämä ohjelmisto on siis siirrettävissä veloituksetta kohteina oleviin asennuskoneisiin.



Kuva 5.17 Esimerkki Visual Basic: ssa käytettävistä objekteista ja ns. robotbox – objektin ominaisuusikkuna avattuna.

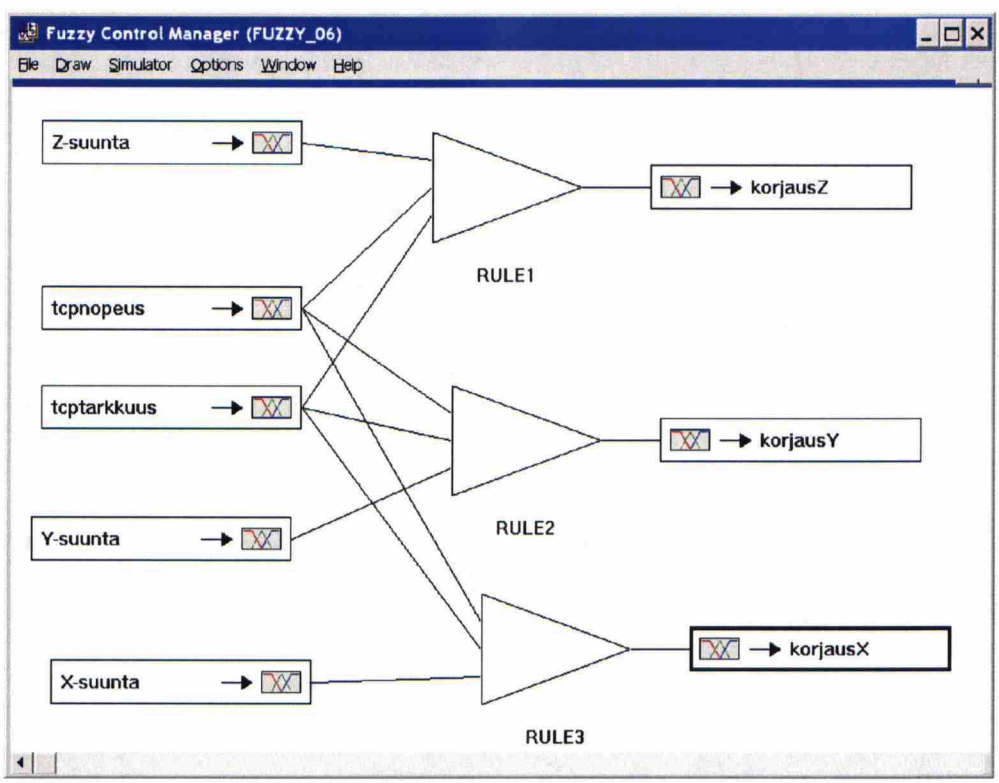
5.5.2 Omron CX – Programmer ja Fuzzy – manager ohjelmistot

CX-Programmer -ohjelma on kehitetty CS1- ja CJ1-logiikoiden ohjelmointiohjelmaksi, mutta se tukee myös lähes kaikkia Omronin C/CV-logiikoita. Yksi projekti voi sisältää useita logiikoita ja ohjelmointitapoina on relekaavio ja käskylista. Sen avulla voidaan toteuttaa myös yhteisten muuttujien ja laitemäärittelyjen hyödyntämistä muiden CX-Serveriä käyttävien ohjelmistojen kesken. Ohjauksen testausmahdollisuus CS/CJ-logiikoille voidaan toteuttaa erikseen hankittavalla CX-Simulator -ympäristöllä. CX-Programmer käyttää siis CX-Server –kommunikointiohjelmistoa. Omronin CX-Server -ohjelmistopaketin mukana tulevaa OPC-serveriä hyödynnettiin käyttöliittymän luonnissa, jolloin saadaan myös esimerkiksi Ethernet-yhteys PC:ltä ohjelmoitavaan logiikkaan, joka puolestaan kommunikoi robotin S4C ohjaimen kanssa.
CX-Programmer ohjelmistolla on luotu ohjelmoitavan logiikan ohjelma, jossa ohjataan sumean säädön avulla robotin liikeradan toteutuksen mittausta ja robotille annettavaa ”korjauskäskyä”.

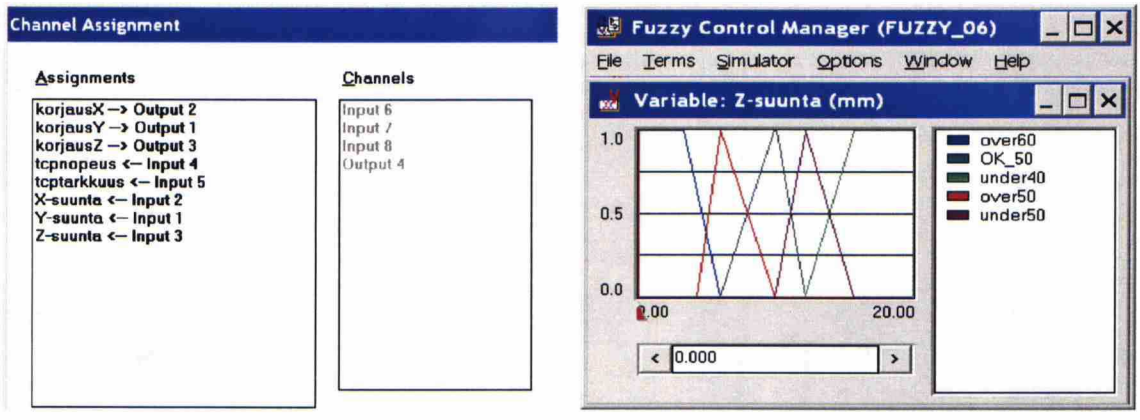


Kuva 5.18 CX -Programmer -ohjelman näkymä ja oikealla ohjelman editointi – ikkuna.

Sumean logiikan tietämyskannan luontiin, sen lataamiseen logiikkaan ja yksikön toimintojen seurantaan käytettiin Fuzzy-manager tukiohjelmistoa.



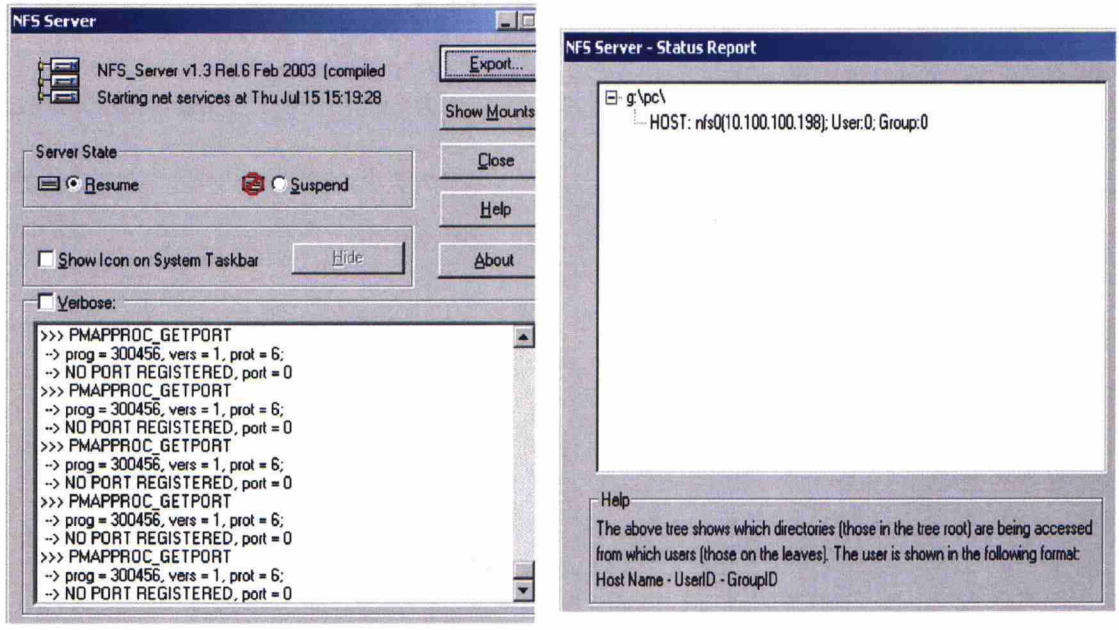
Kuva 5.19 Laserantureiden käytössä toteutettu sumeasäätö, vasemmalla muuttujat, keskellä säännöt ja oikealla päättelyn tuloksena saatavat selkeytykset.



Kuva 5.20 Sumean säädön muuttujien konfigurointi – ikkuna sekä oikealla sanallisten muuttujien määrittelyikkuna.

5.5.3 Microsoft Excel, Visual Basic 6 ja Network File Server (NFS)

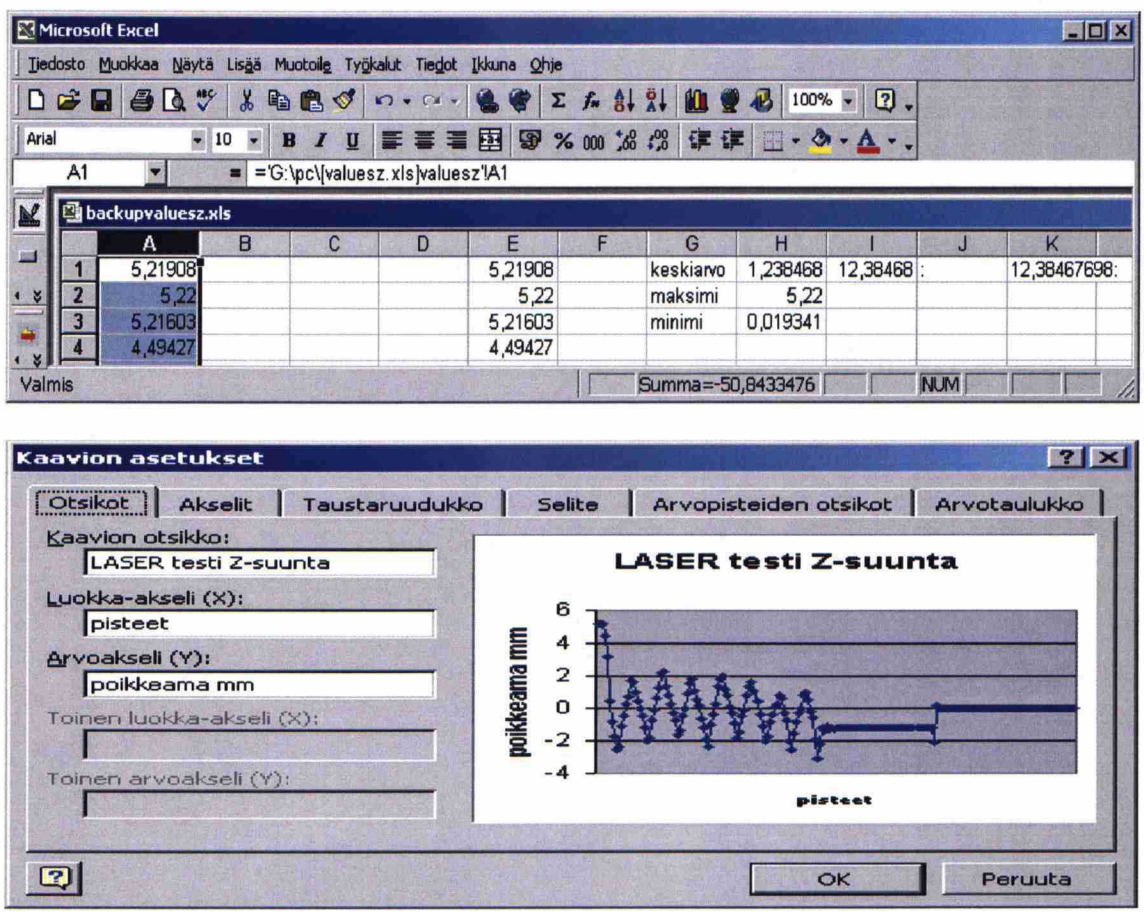
Päätietokoneeseen asennettiin NFS-ohjelmistot, joista tarvitaan sekä NFS-Server ja NFS-Client. NFS-Server mahdollistaa UNIX-pohjaisen levyasemajaon, jolloin robotti voi käyttää tätä asemaa tiedon tallennus- ja siirtopaikkana. Jaetulla levyasemalla oleviin MS Excelin taulukoihin tallennetaan jokaisen työkierron jälkeen laser- ja voima-antureiden antamat poikkeama-arvot verrattuna robotin oikeaan liikerataan, minkä jälkeen voidaan käyttää Excelin omia funktioita niiden käsittelemiseen ja palauttaa saadut arvot robotille. Visual Basic 6.0 -ohjelmistolla luotiin käyttöliittymä, joka on esitelty myöhemmin kohdassa 5.6.



Kuva 5.21 NFS-Serverin hallinta- ja monitorointi-ikkuna ja oikealla nähdään, että jaettu levyasema on g:\pc\ ja siihen on linkitettynä HOST, jonka IP – osoite on 10.100.100.198 (robotti), jonka UserID ja GroupID on 0. Robotille on NFS-Client ohjelmalla määritelty tähän jaettuun levyasemaan kirjoitus- ja lukuoikeudet.

NFS-Serverin käytön yhteydessä on myös robotin järjestelmäparametritietoihin tehtävä seuraavat asetukset:

- tyyppi: NFS
- tiedonsiirtoprotokolla: TCP/IP
- serverin osoite: 10.100.100.152
- luotettuyhteys: ei
- paikallinen polku: tietokone
- serverin polku: /g/pc
- USERID: 0
- GROUPID: 0
- näytetäänkö ohjausyksikössä: kyllä



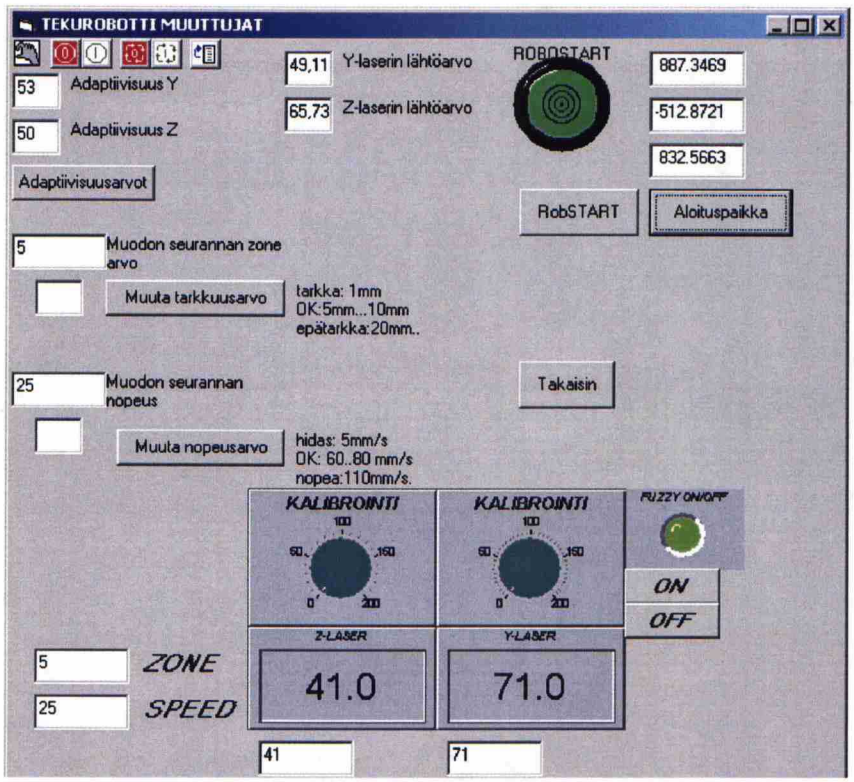
Kuva 5.22 Ylemmässä kuvassa nähdään laserin mittaamia poikkeama-arvoja z-suunnassa sarakkeessa A ja sarakkeeseen H on laskettu keskiarvo sekä etsitty minimi- ja maksimiarvot. Sarakkeessa K oleva arvo on robotille palautettava adaptiivisuuden korjauskerroin, jonka lähettämiseksi robotille tarvitaan myös VBA – koodia. Myös kaavioiden muodostaminen laserin mitta-arvoista on mahdollista (alempi kuva).

Tarvittava VBA – koodi Excel-laskentalomakkeelle:

```
Private Sub CommandButton1_Click ()  {Sama koodi myös: Private Sub Worksheet_Calculate ()}
    Dim fso, txtfile, num
    Set fso = CreateObject ("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile ("g:\pc\numz.txt", True)
    txtfile.Write (Cells (1, 11))
    txtfile.Close
End Sub
```

5.6 KÄYTTÖLIITTYMÄ

Päätietokoneelle luotiin myös käyttöliittymä helpottamaan koeajojen suoritusta ja tuloksien esittämiseksi.



Kuva 5.23 Kuvassa nähdään eräs tutkimuksissa käytetyistä käyttöliittymistä. Liite 4 sisältää toisen esimerkin käytetyistä käyttöliittymistä.

Käyttöliittymän ja robotin välillä on Ethernet-yhteys kommunikointia varten. Käyttöliittymän ja sumean säädön välillä on RS 232-yhteys ja logiikalle yhteys käyttöliittymästä on toteutettu ns. toolbus – väylällä (19 200 bit/s).
Käyttöliittymällä on painikkeet robottiohjelman ja sumean säädön käynnistämiseksi. Robotin liikkeiden parametriarvoja kuten liikenopeusarvoa ja tarkkuusparametriarvoa voidaan myös muuttaa käyttöliittymän kautta. Robotin muodon seurannan radan aloituskoordinaatit saadaan käyttöliittymälle näkyviin samoin kuin laserantureiden ja vastaavasti voima-antureiden mitta-arvot.

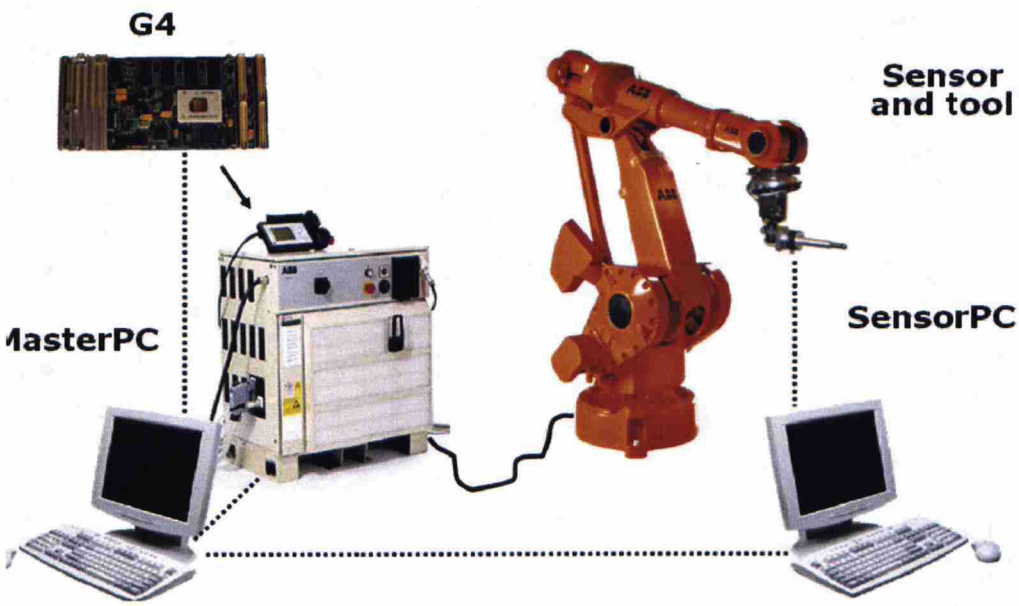
5.7 REFERENSSITUTKIMUS I [9]

5.7.1 Johdanto

Ruotsalaisissa Lundin Tekniikan laitoksessa (Lund Institute of Technology) ja Linköpingin Teknillisessä korkeakoulussa (Linköping Institute of Technology) on kehitetty robottijärjestelmä, joka kykenee nopeaan anturitiedon päivitykseen takaisinkytkentä operaatiossa sekä ennen kaikkea robottiohjain kykenee päivittämään

uutta robotin rataa riittävän nopeasti. Robottien puutteenahan on ollut aistien puuttuminen tai niiden vajanaiset ominaisuudet. Voima ja näkö ovat kaikkein hyödyllisimmät robottijärjestelmän aistit. Paljon tutkimustyötä on tehty reaaliaikaisen anturoinnin ja ohjauksen kehittämisessä robotteihin. Vaikeutena on ollut kuitenkin se, että robottien valmistajat ovat varustaneet robotit melko rajoittuneella tuella tähän erittäin vaativaan toimintaan. Normaali anturitiedon takaisinkytkentä taajuus on ollut roboteissa 10 Hz (0.1 s) ja parhaimmillaankin 20 Hz (0.05s), jotka molemmat arvot ovat täysin riittämättömiä todelliseen reaaliaikaiseen robotin liikeradan luomiseen. Tämä jäljempänä esiteltävä monimutkainen järjestelmä kykenee reaaliaikaiseen 250 Hz:n anturitiedon päivittämiseen robottiohjaimelle.

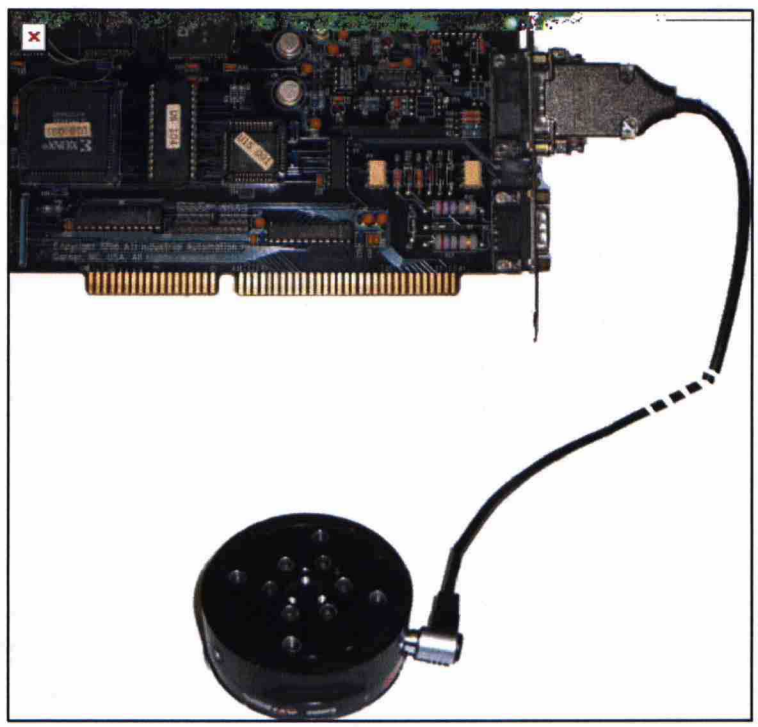
5.7.2 Järjestelmän yleiskuvaus ja tulokset



Kuva 5.24 Yleiskuva järjestelmästä, jossa robottiohjaimeen on liitetty lisäkortti.[9]

Järjestelmän kuuluu kaksi erillistä PC-tietokonetta, joista toinen on ns. masterkone ja toisen PCI -väylään on liitetty ATI/ FT -voimasovitin. Robottina on vastaava ABB IRB 4400 kuin tutkimustyössäkin on ollut, mutta robottiohjain on uudempi eli S4C+ -ohjain, johon on myös ohjelmien osalta mahdollista lisätä ExtRapid-lisäoptio. Oleellisin osa on lisätietokonekortti G4, jossa on oma muisti ja prosessori. Erona on myös alkuperäisen RAM-tyyppisen levymuistin korvaaminen Flash tyyppisellä levymuistilla.
Robottiin tarkoitettuja valmiita kuuden vapausasteen voima-anturisovittimia on erilaisissa tutkimuksissa käytetty kahdenlaisia, joista toinen on kauppanimeltään JR3, jonka vahvistinkortti on liitetty suoraan robottiohjaimen S4Cplus PCI-väylään sekä toinen ATI Gamma Force / Torque- anturi, joka on nähtävissä seuraavalla sivulla olevassa kuvassa (kuva 5.25). ATI:n anturisovittimen kortti on liitetty vastaavasti PC:n PCI – tai ISA – väylään.

Tässä vertailututkimustyössä on käytetty ISA-väylään asennettavaa mallia, minkä vuoksi on jouduttu käyttämään erillistä anturitietokonetta.



Kuva 5.25 ATI F/T – anturi ja tietokoneeseen asennettava lisäkortti. [9]

Tässä vertailututkimuksessa käytetty voima-anturi oli mallia ATI Gamma voima /vääntömomentti 130 – 10 anturi, joka voi mitata voimaa noin 8 kHz taajuudella ja maksimi voimat ovat Fz= 400 N, Fx ja Fy= 130 N sekä vääntömomentit Tx, Ty ja Tz= 10 Nm.
Tutkimuksen yhteenvedossa todetaan, että tällainen järjestelmä mahdollistaa reaaliaikaisen robottiradan muodostamisen nyt huomattavasti paremmin verrattuna aikaisemmin hyvin huonosti toimineeseen reaaliaikaisen robotin ohjaukseen antureilta tulevien viestin perusteella ja näin muodostettaessa samanaikaisesti robotin liikerataa. Sellaisissa käytännön sovelluksissa kuten hiominen, poraaminen ja jäysteen poisto vaaditaan robotilta juuri adaptiivista liikkeen suorittamista. Suoritetuissa kokeissa ABB ja KUKA roboteilla on saavutettu jopa 50 mikrometrin tarkkuuksia nopealla takaisinkytkennällä ja metrologisilla antureilla.

5.8 REFERENSSITUTKIMUS II [10]

5.8.1 Johdanto

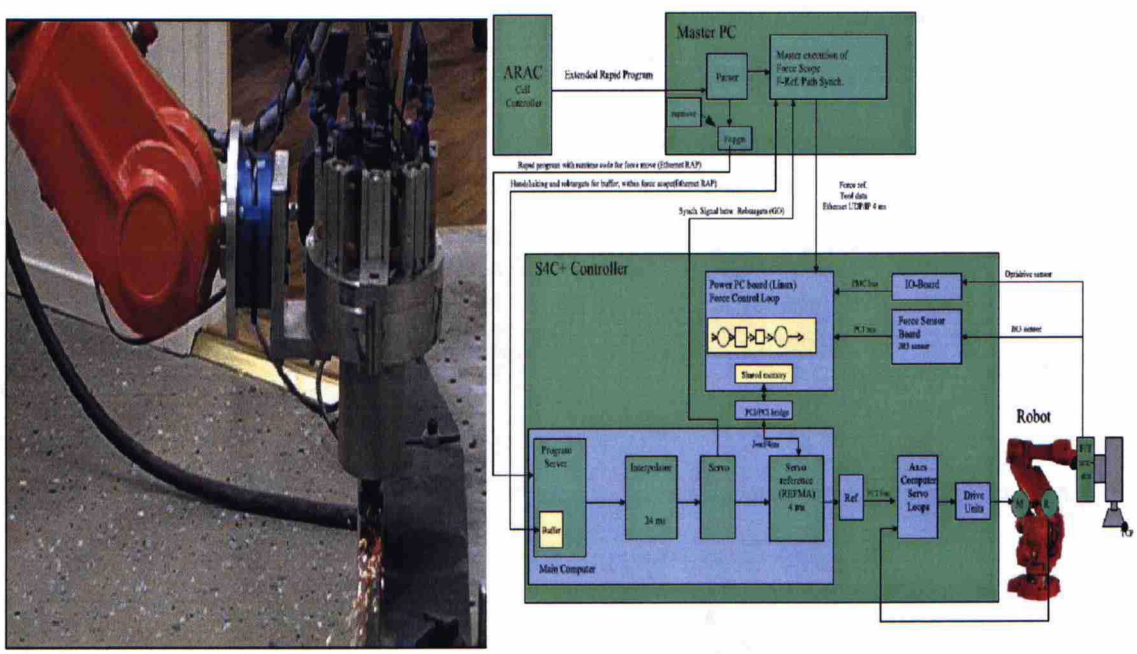
Monta lupaavaa robotiikan tutkimustyötä on peräisin jo 1970-luvun lopulta ja 1980-luvun alulta. Suorakulmaisen koordinaatiston mukaista voimaohjausta on testattu kehittyneessä liikkeen ohjauksessa jo 20 vuoden ajan. Robottien mukautumista

muuttuviin ympäristöolosuhteisiin radan muodostamisen osalta on yritetty parantaa erilaisilla ulkoisilla antureilla. Pyrkimyksenä on saavuttaa todellinen reaaliaikainen robottiradan laskenta ulkoisen anturitiedon perusteella.

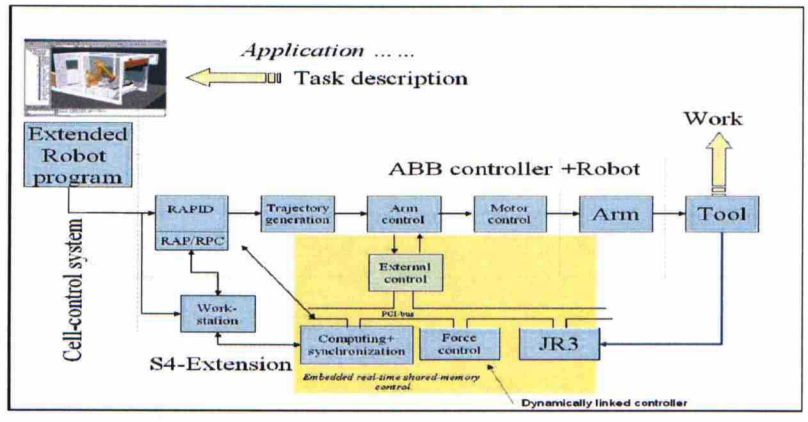
Nykyaikaiset robotit omaavat kyvyn liittää I/O:t Ethernetin tai erilaisten väylätekniikoiden avulla. Kriittinen asia ulkoisen ohjauksen takaisin kytkennän onnistumiselle on käytettävissä olevan tiedonsiirron kaistan leveys. Takaisinkytkennän nopeutta on yleensä testattu yliopistomaailmassa perinteisellä PUMA 560 – käsivarsiroboteilla (**P**rogrammable **U**niversal **M**achine **A**rm). Mutta nykyisten kaupallisten robottien servomoottorien ohjauksen säädön vaikuttamismahdollisuudet ovat huomattavasti rajoitetumpia, koska samanlaista täysin avointa pääsyä vaikuttamaan suoraan servomoottorien säätöalgoritmeihin ei ole olemassa samalla lailla kuin PUMA560 - robottien yhteydessä on ollut.
Tässä toisessa vertailuprojektissa on käytetty myös ABB S4C+ -ohjainta, joka ei ole avoin järjestelmä, mutta sen sisäisen ohjaimen kehittämiseen on jätetty muutamia mahdollisia ominaisuuksia.

5.8.2 Järjestelmän yleiskuvaus ja tulokset

Voima-ohjausta testattiin hionta- ja jäysteen poistosovelluksissa. Projektin lopputuloksena kehitetyn anturikäyttöliittymän uskotaan olevan ainutlaatuinen, koska se on yhdistelmä jaetun muistialueen liittämisestä sisäiseen liikkeen ohjaukseen, mikä mahdollistaa nopean anturitiedon vaikutuksen säätöön. Antureita voidaan käsitellä sekä reaaliaikaisesti niin ohjelman osalta että laitteiston osalta käyttämällä käyttöjärjestelmän moduuleita.



Kuva 5.26 Vasemmalla nähdään voima - anturisovitin kiinnitettynä robotin laippaan ja jäysteen poiston testausta. Oikealla on layout-kuva järjestelmän rakenteesta. [10]



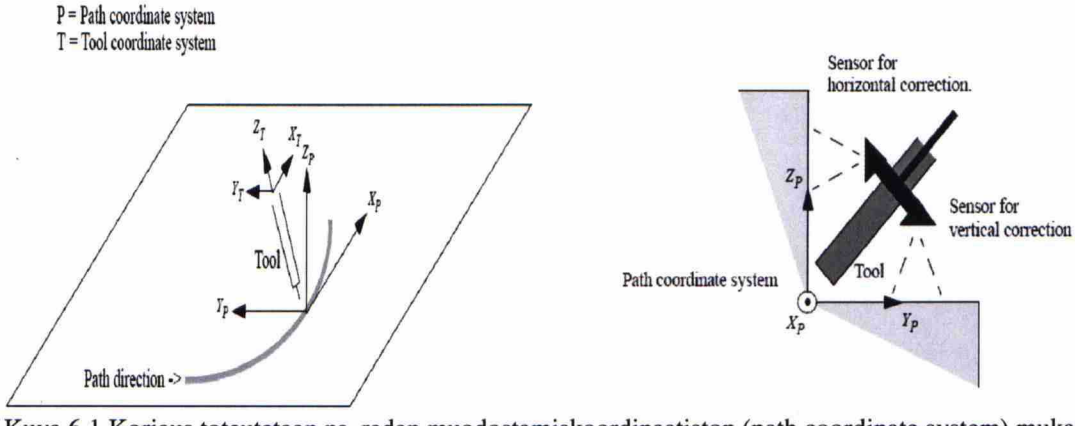
Kuva 5.27 Lohkokaaviokuva laajennetusta ABB S4C+ ohjaimesta. [10]

6. TOTEUTETUT RATKAISUT ja KOEAJOEN TULOKSET

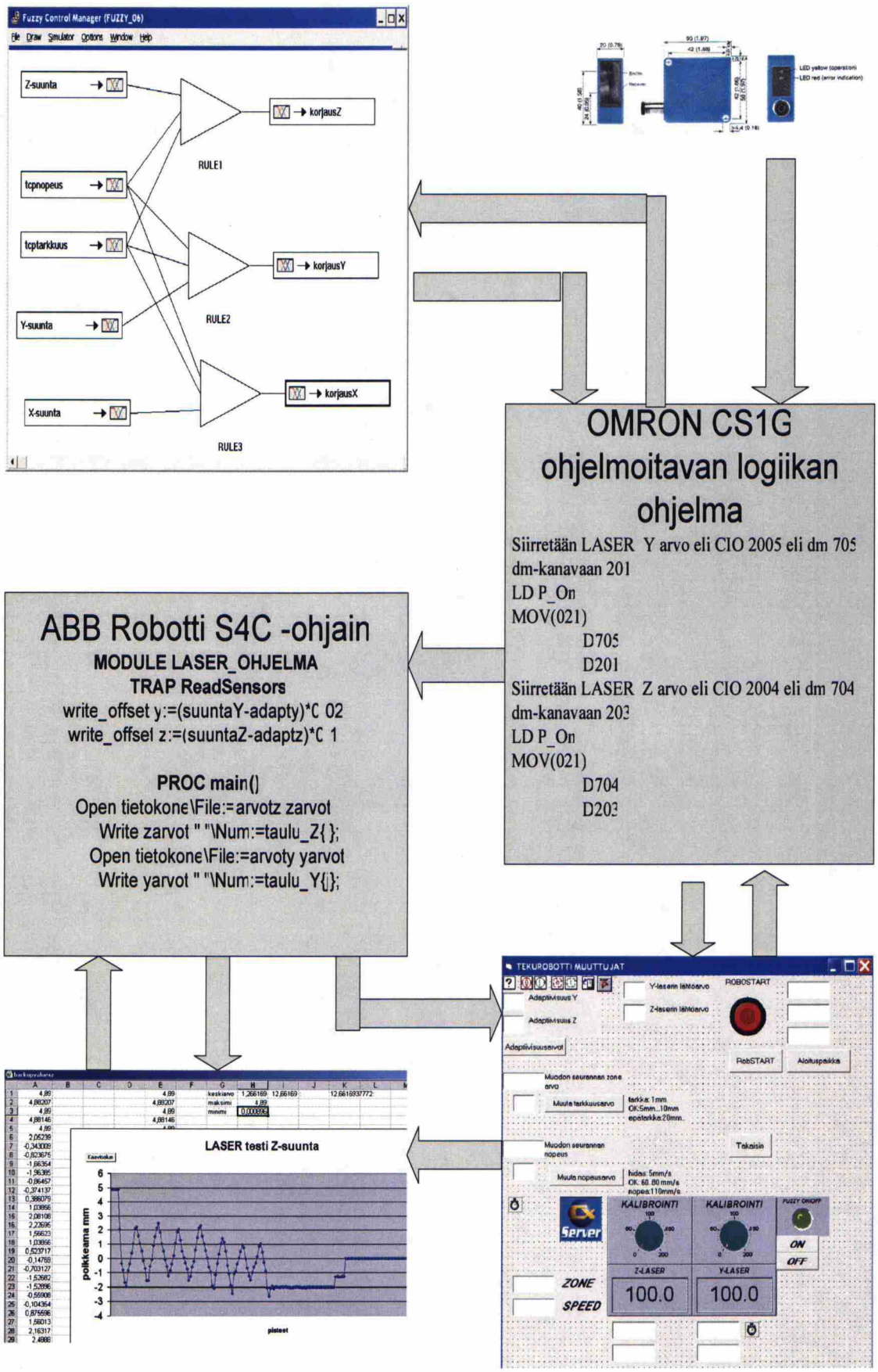
6.1 LASERANTUREIDEN KÄYTTÖ

6.1.1 Johdanto

Robotin adaptiivisen radan korjauksen toteutuksessa voidaan käyttää ns. contour tracking toimintaa. Tämän toiminnon käyttö S4C-ohjaimen yhteydessä vaatii Advanced Motion -option asentamista robotin käyttöjärjestelmään. Radan korjaus toteutetaan radan muodostamiskoordinaatiston suhteen (path coordinate system). Robotin liikkuessa ennalta määriteltyjen alkupisteen ja loppupisteen välillä suoritetaan radan korjaus välittömästi anturitiedon perusteella. Tämän korjauksen toteutuksessa voidaan käyttää ohjelmallisesti kahta menetelmää, joko ns. traproutines eli keskeytysaliohjelman käyttöä tai multitasking eli rinnakkaisohjelmien samanaikaista käyttöä. Robotin rata muodostuu yleensä peruskoordinaatiston mukaisesti, mutta käytettäessä ns. kehyksen siirtoja muodostetaan rata määritellyn kehyksen mukaisesti suhteessa peruskoordinaatistoon ja tästä käytetään nimitystä radan muodostamiskoordinaatisto.



Kuva 6.1 Korjaus toteutetaan ns. radan muodostamiskoordinaatiston (path coordinate system) mukaisesti, eikä työkalukoordinaatiston (tool coordinate system) mukaan. Oikealla esimerkki anturien asennuksesta suoritettaessa radan korjausta z - ja y -suunnissa. [18]



Kuva 6.2 Kaaviokuva toteutetussa ratkaisussa käytettyjen ohjelmistojen tietovirroista.

Laserantureiden viestit luetaan ohjelmoitavan logiikan analogiatuloihin, joista ne otetaan käyttöön sumean logiikan sanallisiksi muuttujiksi. Sumean logiikan päättelyn lopputulos lähetetään ohjelmoitavan logiikan analogialähdöillä robotin analogiatuloille. Robotin liikkeen korjaus toteutetaan ja mitatut poikkeamat tavoitellusta robotin liikeradasta lähetetään käyttöliittymälle, josta voidaan tehdä myös parametriasetuksia kuten robotin tarkkuusalueen muutokset ja liikenopeuden muutokset.

6.1.2 Robottiohjelmat

Kehittyneen robotin liikkeen ohjauksen (Advanced Motion) käytössä on useita erilaisia käskyjä ja datatietoja, joita joudutaan käyttämään ja määrittelemään. Samoin joudutaan tekemään myös keskeytysaliohjelman (TrapRoutines) kanssa. Seuraavaksi kuvataan ensin näiden esittely. Aliohjelman pienin kutsuväli on 0,05 s, jolloin paikoitustietojen ohjaus anturiarvojen perusteella saavuttaa 20 Hz taajuuden.

Tarvittavat datatiedot:

Corrdescr (Correction generator descriptor) eli ns. korjauskehittimen käytöllä lisätään geometrinen poikkeama robotin liikerataan.

```
VAR corrdescr corrdescr_Z;
VAR corrdescr corrdescr_Y;
VAR corrdescr corrdescr_X;
```

Muuttujatyyppi intnum määrittelee keskeytystulon, johon voidaan liittää tietyn keskeytysaliohjelman kutsu.

```
VAR intnum intno1;
```

Tarvittavat käskyt:

Liityntä määriteltyyn korjauskehittimeen(Corrdescr) tapahtuu CorrCon -käskyllä.

```
CorrCon corrdescr_Z;
CorrCon corrdescr_Y;
CorrCon corrdescr_X;
```

Liitynnän katkaisu tapahtuu CorrDiscon -käskyllä. Kaikki yhteydet voidaan katkaista myös yhdellä käskyllä eli CorrClear katkaisee samalla kertaa kaikki liitynnät.

```
CorrDiscon corrdescr_Z;
CorrDiscon corrdescr_Y;
CorrDiscon corrdescr_X;
CorrClear;
```

Korjauksen kirjoitus eli toteutus suoritetaan CorrWrite -käskyllä.

```
CorrWrite corrdescr_Y, write_offset;
```

Funktiolla CorrRead voidaan lukea kaikki toteutetut korjausarvot.

```
total_offset:=CorrRead();
```

CONNECT -käskyllä kytketään tietty keskeytystulo (esim.intno1) annettuun keskeytysaliohjelmaan (esim. ReadSensors).

```
CONNECT intno1 WITH ReadSensors;
```

ITimer -käskyllä määritellään keskeytysaliohjelman kutsuväli ja parametrilla [\Single] voidaan kutsu määritellä tapatuvaksi vain yhden kerran.

```
ITimer 60, intno1; (kutsutaan 60 s:n väliajoin aliohjelmaa)
ITimer\Single, 0.05,intno1; (kutsutaan yhden kerran 0.05 s:n kuluttua)
```

Toteutetun robottiohjelman selostus kuvataan liitteessä 1.

6.1.3 Ohjelmoitavan logiikan ja sumean säädön toteutus

Ohjelmoitavaan logiikkaan on tehtävä ohjausosat robotille lähetettävää dataa varten, sumean säädön vaatimia määrittelyjä varten sekä Visual Basic ohjelmointiympäristöllä toteutettua käyttöliittymää varten.

CX-Programmer – ohjelmistolla tehty ohjelma ohjelmoitavaan logiikkaan:

Robotin analogiatulojen arvot ohjelmoitavalta logiikalta sumean päättelyn tuloksina:
Kanavaan 2020(20n0) ilmoitetaan analogiatulojen määrä eli 5 kpl.

```
LD P_On
MOV
    #5
    2020
```

Kanavaan 2021 ilmoitetaan sen sanan osoite, joka on ensimmäinen sumea tulon analogiatulo. (Luku 0 ilmentää datamemory-aluetta ja sieltä sana 201 on siis ensimmäinen sumea tulo.)

```
LD P_On
MOV
    #0201
    2021
```

Siirretään datamuistissa 704 oleva LASER _Z arvo kolmanteen sumeaan tuloon (D203)

```
LD P_On
MOV
    D704
    D203
```

Siirretään datamuistissa 705 oleva LASER _Y arvo ensimmäiseen sumeaan tuloon (D201)

```
LD P_On
MOV
    D705
    D201
```

Kanavaan eli sanaan 2022 ilmoitetaan analogialähtöjen määrä eli nyt 4 kpl.

```
LD P_On
MOV
    #4
    2022
```

Kanavaan 2023 ilmoitetaan sen sanan osoite, joka on ensimmäinen sumea lähtö. (Luku 0 ilmentää datamemory-aluetta ja sieltä sana 110 on siis ensimmäinen sumea lähtö.)

```
LD P_On
MOV
    #110
    2023
```

Ensimmäisen sumean lähdön siirto robotin analogiatuloihin (vastaavasti kaikki muut myös).

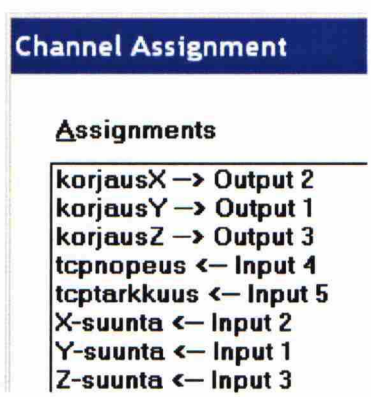
```
LD P_On
MOV
    D110
     2010
```

Sumean säädön käynnistys tapahtuu Visual Basic – käyttöliittymältä.
Aloitetaan tai lopetetaan sumeaohjaus

```
LD 30.01
LDNOT 30.01
KEEP
    30.00
LD 30.00
OUT 2020.15
```

Sumean säädön säännöt:

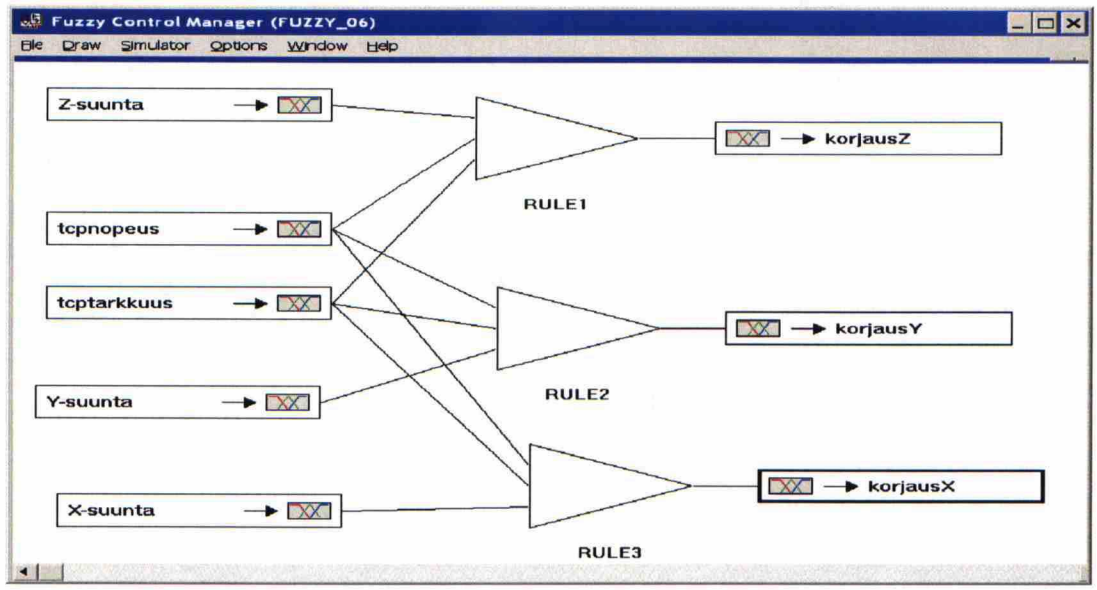
Sumeita muuttujia on siis yhteensä viisi kappaletta, joista kolme liittyy robotin koordinaattiakseleihin (x, y ja z) sekä kaksi robotin liikenopeuteen ja paikoitustarkkuuteen. Sääntöjä on kolme kappaletta, joissa kussakin on yksi akselimuuttuja ja aina molemmat robotin liikkeen ohjaukseen tarkoitetut muuttujat. Sääntöjen tuloksena on luonnollisesti kolme päättelyosaa, joista saadaan robotin analogiatulojen kautta tarvittavat liikekorjausarvot.



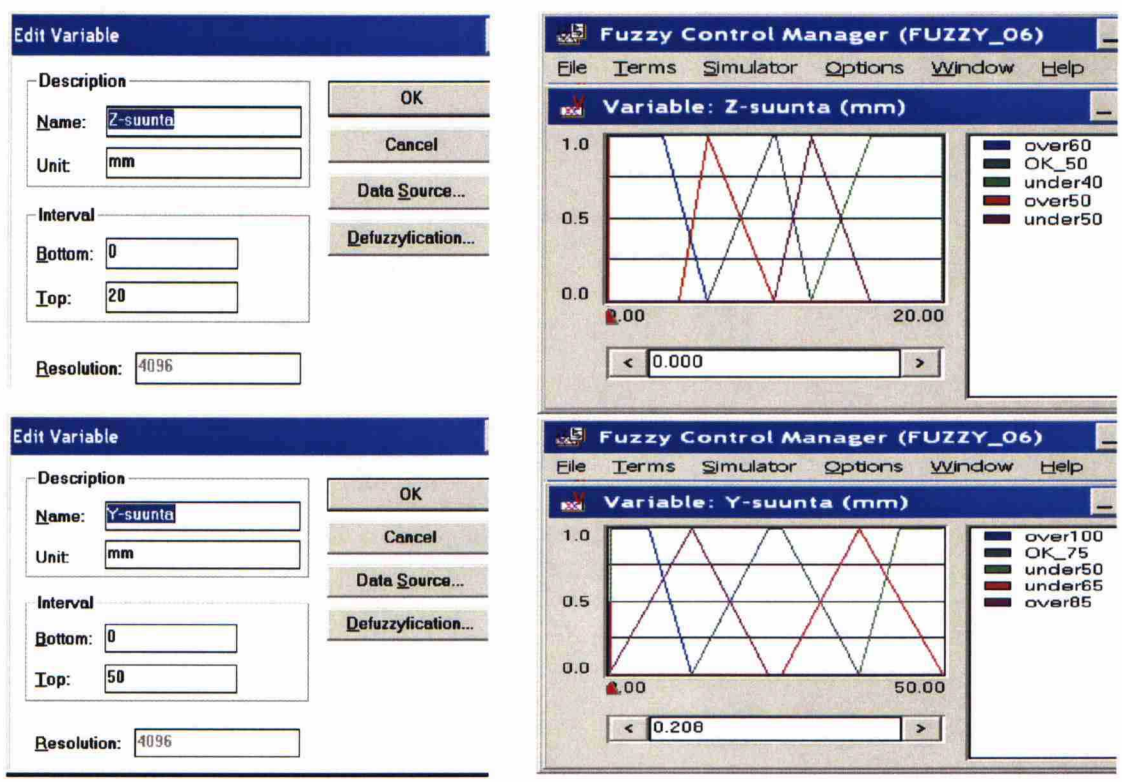
Output1 → robotin analogiatulo 1
Output2 → robotin analogiatulo 2
Output3 → robotin analogiatulo 3

Input1 ← laserY
Input 2 ← varalla tai X-suuntainen voima
Input 3 ←laserZ
Input 4← robotin liikenopeus
Input 5 ← robotin paikoitustarkkuus

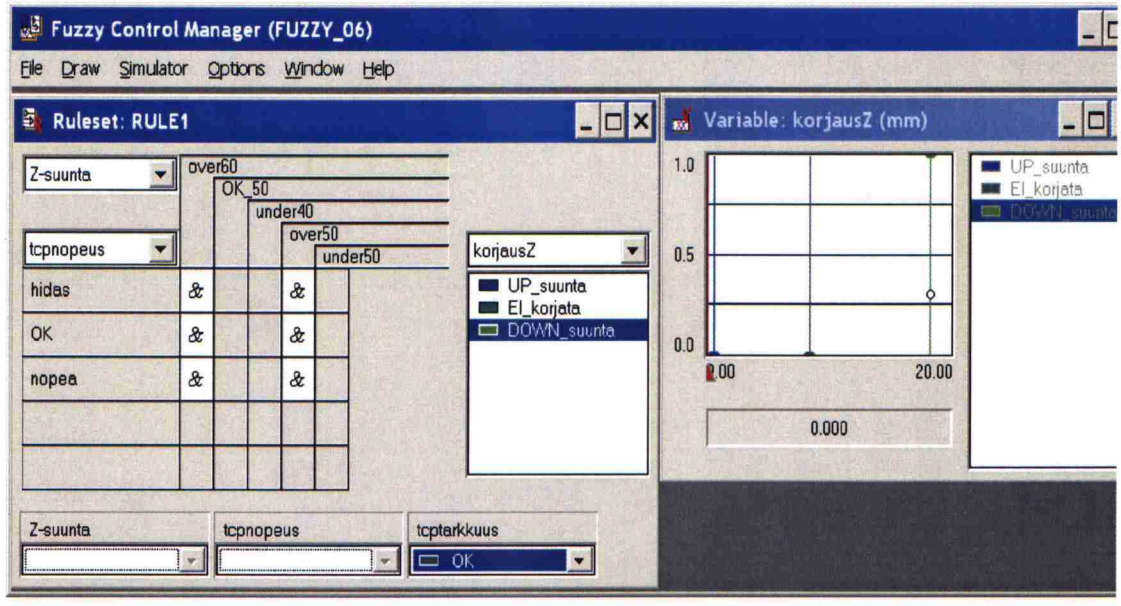
Kuva 6.3 Sumean säädön muuttujat ja selkeytykset.



Kuva 6.4 Sumean säädön sääntöjen ja päättelyjen graafinen näkymä.



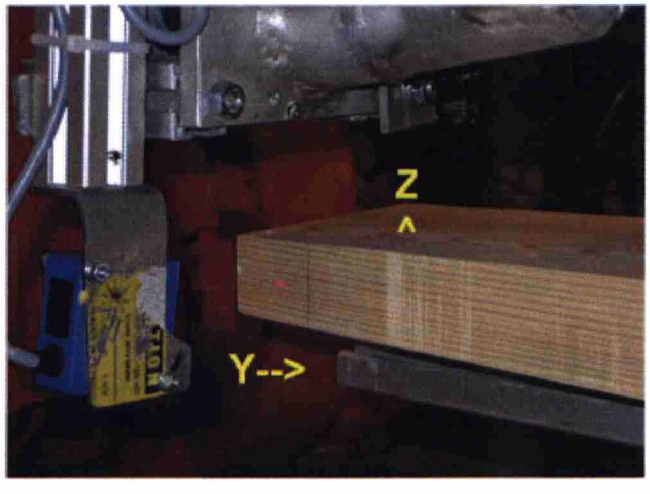
Kuva 6.5 Ylhäällä on kuvattu sumean säädön muuttuja Z – suunta, mittausväli on 20 mm ja tämä on jaettu viiteen sanalliseen väliin sekä alhaalla vastaavasti sumean säädön muuttuja Y – suunta, mittausväli on 50 mm ja myös tämä on jaettu viiteen sanalliseen väliin.



Kuva 6.6 Esimerkki sumean säädön sääntöjen muodostamisesta maksimiarvomenetelmällä.

6.1.4 Koeajon tulokset ja johtopäätökset

Aluksi ennen varsinaisten koeajojen suorittamista suoritettiin pidemmän aikaa testiajoja rakennettaessa järjestelmää. Näiden testiajojen jälkeen päädyttiin siihen, että sopivin paikoitustarkkuus (zone) on 5...10 mm ja liikenopeus 50 mm/s. Seuraavaksi testattiin sumean säädön vaikutusta radan muodostamisessa.

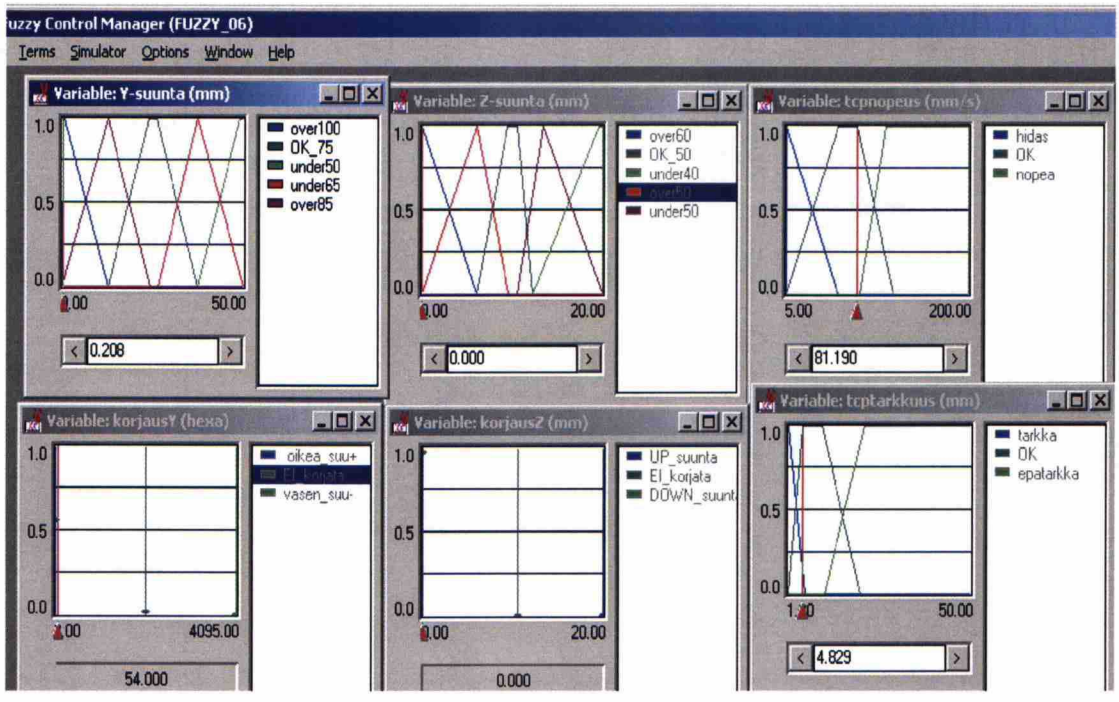


Kuva 6.7 Koekappaleena käytettiin y- ja z-suunnassa koneistettua viistomaista kappaletta.

Koeajot (TrapRoutines – ohjelmalla):

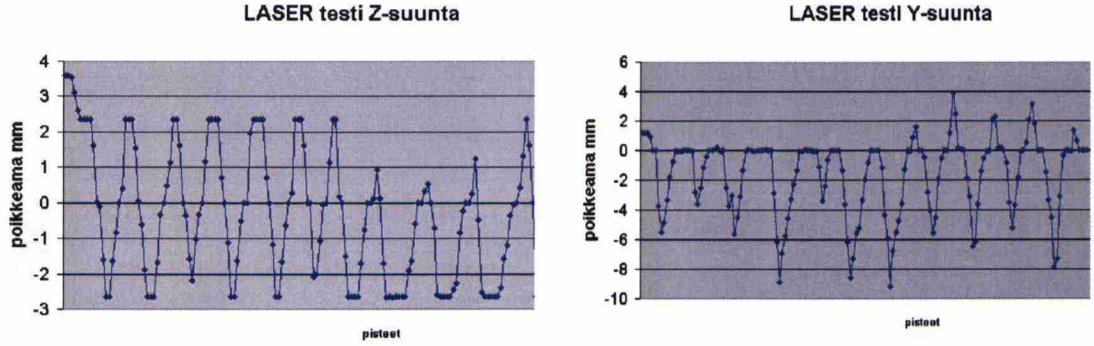
Testi 1:

Alla olevasta kuvasta nähdään sumeiden muuttujien ja päättelyiden lähtötilanne. Esimerkki selkeytyssäännöistä on esitelty kuvassa 6.6.



Kuva 6.8 Sumeat muuttujat Y-suunta ja Z -suunta ja niiden alapuolella vastaavat korjaukset päättelyn tuloksena. Oikealla kuvassa nähdään myös sumeat muuttujat tcpnopeus (50 mm/s) ja tcptarkkuus (5 mm).

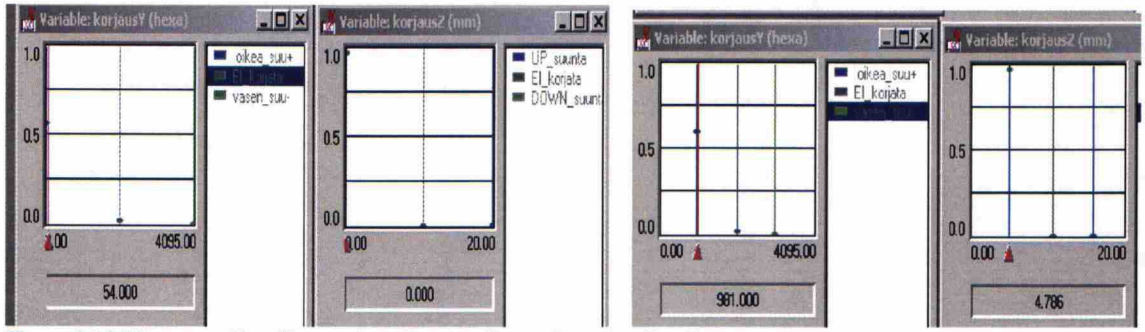
Laserantureilla toteutettu reaaliaikainen radan muodostaminen ja taulukkoon tallennetut poikkeamat todellisesta radasta voidaan lukea Microsoft Excel – ohjelman muodostamista käyristä sekä y- ja z -suunnissa.



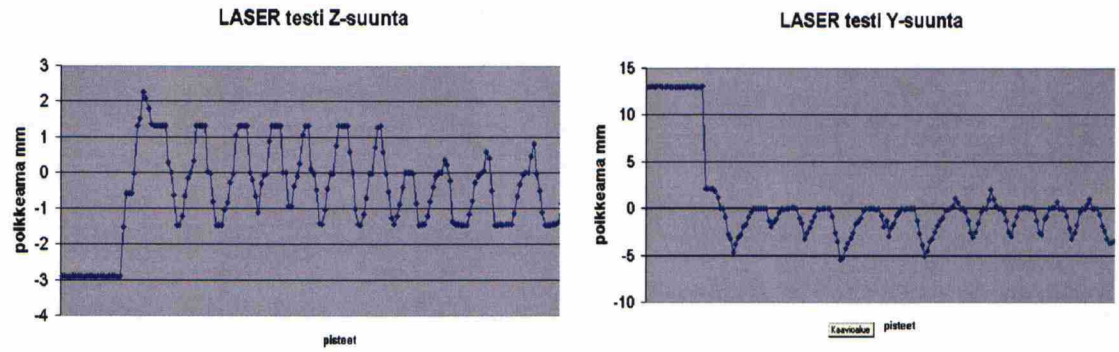
Kuva 6.9 Lasermittauksen tulokset, poikkeama robotin tavoitellusta radasta y- ja z- suunnissa. Kuten käyristä nähdään, niin poikkeamat ovat suuria ja tämä tilanne on tavallaan lähtötilanne.

Testi 2:

Seuraavaksi muutettiin molempien suuntien selkeytyksien vasemman ja oikean puolimmaisia pystyjanoja keskemmälle eli tilannetta, jolloin ei suoriteta robotin liikeradassa korjausta.

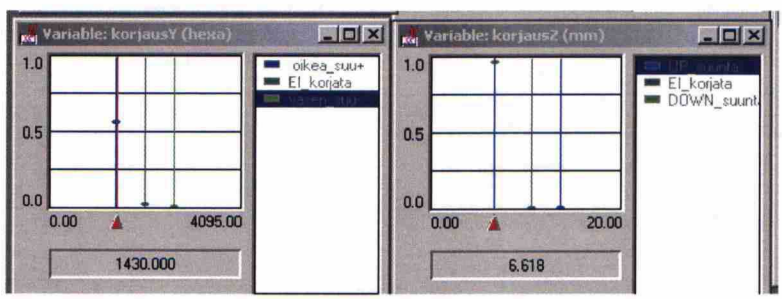


Kuva 6.10 Vasemmalla alkuperäiset (y, z) selkeytykset ja oikealla muutetut.

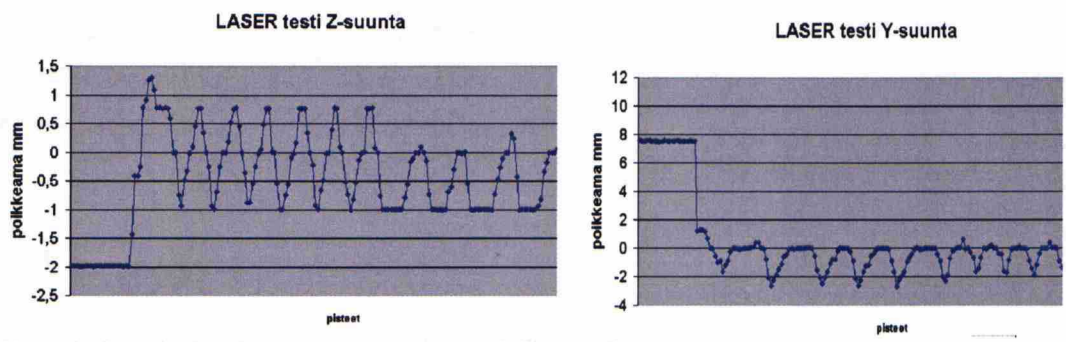


Kuva 6.11 Etenkin z – suunnassa tapahtui liikkeen tasaantumista, - /+2.5 mm vaihtelu tasaantui välille -/+ 1.5 mm.

Seuraavaksi siirrettiin edelleen janoja keskemmälle.



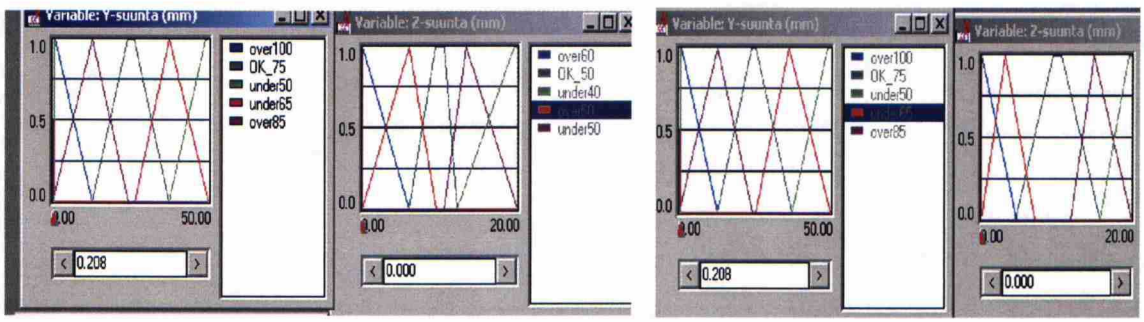
Kuva 6.12 Siirrettiin äärijanoja vieläkin keskeisemmäksi.



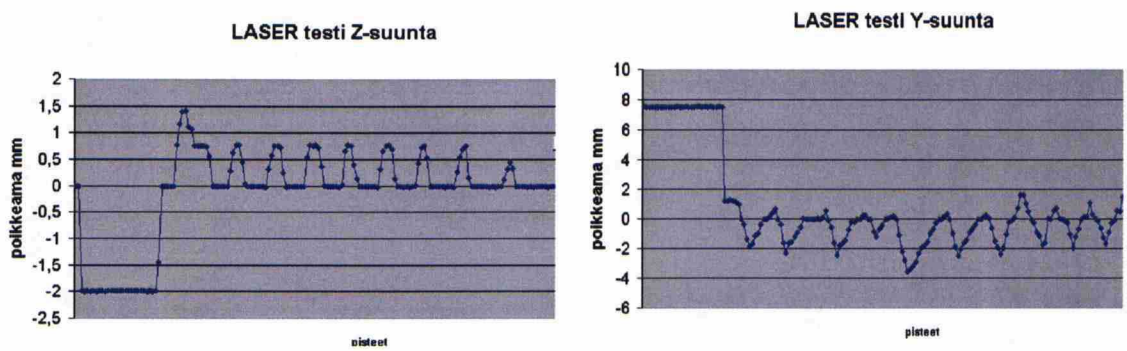
Kuva 6.13 Vaihteluvälit pienenivät edelleen edellisestä tilanteesta.

Testi 3:

Seuraavaksi muutettiin sanallisten muuttujien graafisia käyriä, mutta muuten asetukset pysyivät samoina kuin edellisessä testissä olivat.



Kuva 6.14 Vasemmalla alkuperäiset (y, z) ja oikealla muutetut graafiset käyrät sumeille muuttujille.

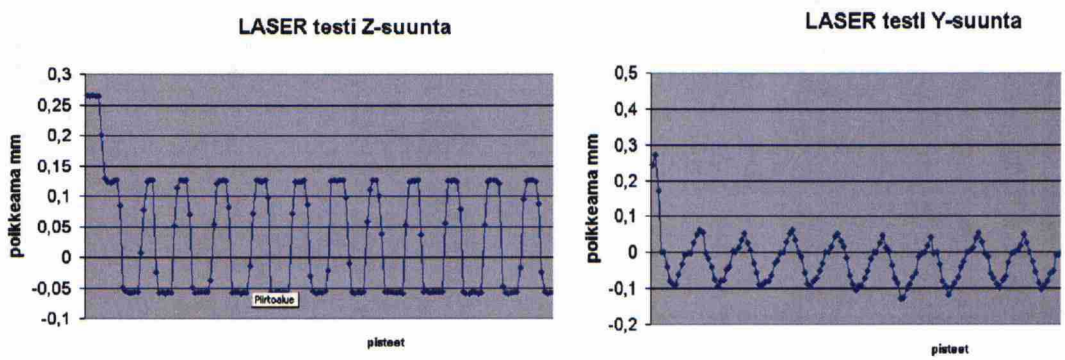


Kuva 6.15 Nyt z – suunnassa lähestyttiin toisessa suunnassa 0 mm:n poikkeamaa oikeasta radasta ja toisessakin suunnassa saavutettiin vain noin 0.5 mm:n virhe.

Testi 4:

Haettaessa parempaa lopputulosta reaaliaikaisen radan muodostamisessa päädyttiin tässä vaiheessa siihen, että kaikki mitä sumealla säädöllä on tehtävissä, on nyt suoritettu ja seuraavaksi hienosäädettiin robotin ohjelmaa ja sen parametreja.

Seuraavaksi muutettiin tcptarkkuus eli zone arvo 5 mm:iin ja robotin liikenopeus alennettiin puoleen eli nopeus oli nyt 25 mm/s. Lisäksi hienosäädettiin robotin ohjelmassa olevia käskyjä write_offset.z:=(suuntaZ-adaptz)****0.02*** ja write_offset.y:= (adapty - suuntaY)****0.02*** write_offset.x:=-(tcpnopeus/20), jolloin tulokset paranivat oleellisesti ja nyt myös y-suunnassa (robottiohjelma on liitteessä 1).



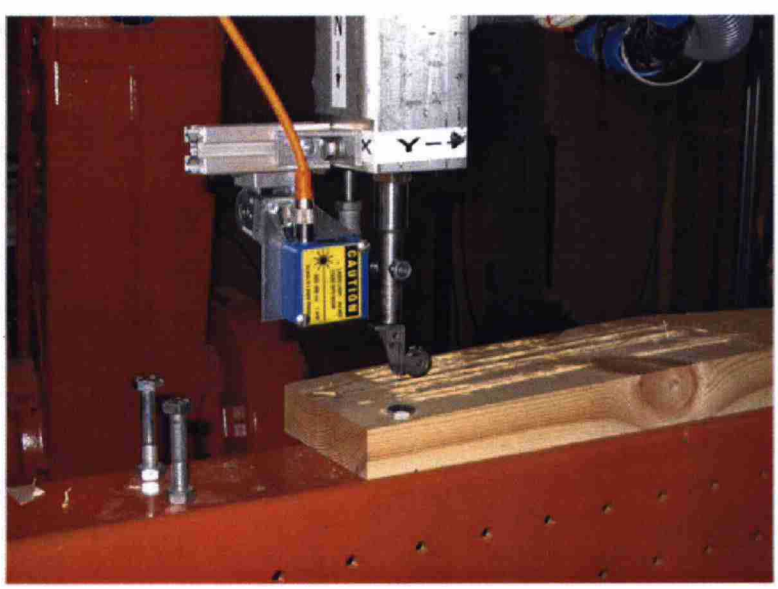
Kuva 6.16 Z – suunnassa lähestyttiin toisessa suunnassa 0.05 mm:n poikkeamaa oikeasta radasta ja toisessakin suunnassa noin. 0.125 mm virhettä. Myös y – suunnassa saavutettiin oleellisesti pienemmät poikkeamat +/- 0.1 mm.

Yhteenveto:
Robotin parametrien osalta liikenopeus ja paikoitustarkkuus vaikuttivat muodon seurannan lopputulokseen. Parhaimmat arvot saavutettiin liikenopeudella, joka oli välillä 25 mm /s ...60 mm/s ja paikoitustarkkuudella (zone – alue) 5 mm. Paikoitustarkkuutta parantamalla (esimerkiksi 1 mm) tai huonontamalla (esimerkiksi 10 mm) poikkeamat oikeasta radasta suurenivat. Luettavan radan korjauksien skaalauskertoimet kuten esimerkiksi *write_offset.z:= (suuntaZ-adaptz)*0.02* vaikuttivat myös oleellisesti. Pienimmät poikkeamat saavutettiin arvolla 0.02. Sumean ohjauksen säädöillä oli kuitenkin kaikkein suurimmat vaikutukset optimoidessa muodon seurannan toteuttamista ja etenkin alussa suoritetuissa säädöissä.

6.2 VOIMA-ANTUREIDEN KÄYTTÖ

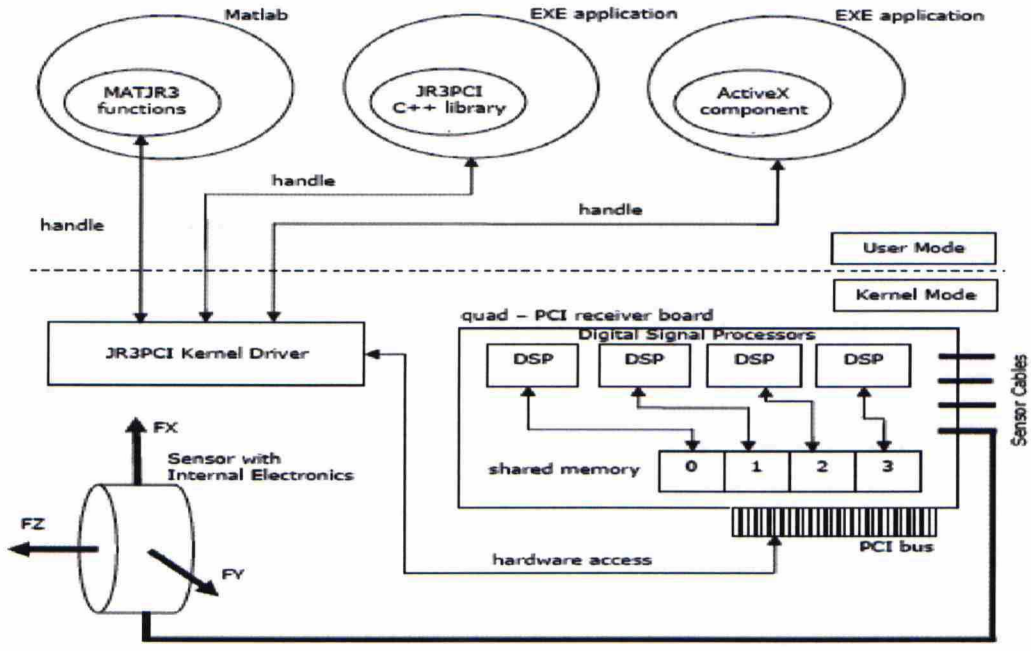
6.2.1 Johdanto

Laserantureilla aikaisemmin tehtyä tutkimusta voidaan tavallaan pitää esitutkimuksena voima-antureilla tehdyille koeajoille. Voima-anturisovitin robotin työkalulaippaan suunniteltiin ja toteutettiin itse, koska markkinoilla olevat ATI- ja J3-voima-anturisovittimet ovat sen verran arvokkaita, ettei niitä ollut mahdollista hankkia. Oma valmiste varustettuna Raute Oy:n valmistamilla punnitusantureilla toimi kuitenkin riittävän hyvin tämän tutkimuksen tarpeita varten. Tosin rajoituksena oli mahdollisuus mitata voima vain kolmen pääakselin suunnassa, jolloin kiertoliikkeiden vääntömomenttien mittausmahdollisuutta ei ollut käytettävissä. Ennen koeajojen suoritusta jouduttiin suorittamaan kalibrointi siten, että anturien antama mittaviesti muutettiin ohjelmoitavan logiikan kautta vastaamaan todellisia Newton voima-arvoja. Tutkimukset aloitettiin ilman sumean säädön apua ja voiman mittaus tapahtui vain z-suunnassa sekä lisäksi testattiin erilaisten robotin systeemiparametriarvojen vaikutusta.



Kuva 6.17 Robotin voima - anturisovittimeen kiinnitettiin laakeripyörä, joka tavallaan toimi työkaluna. Lisäksi lisättiin laseranturi mittaamaan tietyllä voimalla ajetun robottiradan muodostaman uran syvyyttä.

Voiman ja momentin mittaukseen perustuva ohjaus olisi tarpeellinen monissa automaatio- ja robottisovelluksissa. Robottien ohjauksessa hyväksi käytettävä voima-ohjaus voidaan jakaa passiiviseen ja aktiiviseen ohjaukseen. Passiivisessa voimaohjauksessa robotin työkaluun kohdistuva voima saa vaihdella sallitulla etukäteen määritetyllä turvallisella alueella. Aktiivisessa voimaohjauksessa työkaluun kohdistuvat voimat vaikuttavat suoraan ja välittömästi robotin suorittaman tehtävän lopputulokseen. Passiivisessa voimaohjauksessa työkaluun kohdistuvat voimat eivät ole toivottuja, mutta hyväksyttäviä kunhan ne ovat määritellyn alueen sisällä, jolloin tällaisissa sovelluksissa saadaan joustavuutta lisättyä. Sen sijaan aktiivisessa ohjauksessa muuttuvat voima-arvot ohjaavat robotin toimintaa ja ovat siis välttämättömiä koko sovelluksen onnistumisen kannalta.
Portugalialaisen professorin Norbert Pireksen johtamassa tutkimuksessa käytettiin JR3-voima-anturisovitinta, joka koostuu sensoriosasta. Se kykenee osoittamaan vaikuttavat voimat karteesisen koordinaatiston x-, y- ja z-suunnissa. Lisäksi tietokoneeseen on asennettava erillinen vahvistinkortti. Voima-anturin näytteenottotaajuus riittää erinomaisesti, koska se voi olla jopa 8 kHz. Kyseisessä projektissa käytettiin hyväksi myös Matlab-ohjelmistoa, johon tämä PCI-väyläinen DSP (digitaalinen signaalin käsittely) – kortti onnistuttiin liittämään. [16]



Kuva 6.18 Lohkokaaviokuva järjestelmästä, johon on liitetty PCI – väyläinen DSP – sovitinkortti MATLAB – ohjelmiston yhteyteen. [16.]

6.2.2 Robottiohjelmat

Robotin ohjelmat toteutettiin samalla periaatteella kuin aikaisemminkin käytettäessä laserantureita. Ohjelmat on selostettu liitteessä numero kaksi.

6.2.3 Ohjelmoitavan logiikan ja sumean säädön toteutus

Ohjelmoitavan logiikan ohjelma ja käyttöliittymä on selostettu liitteissä kolme ja neljä.

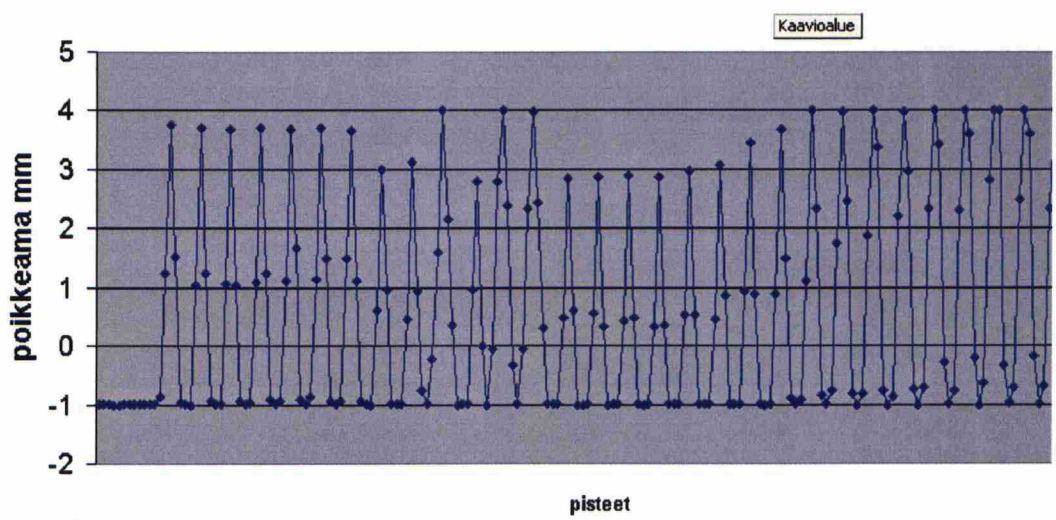
6.2.4 Koeajon tulokset ja johtopäätökset

Koeajot (TrapRoutines – ohjelmalla):

Testi 1:

Voima mitattiin (tavoitearvo 200 N) vain z – suunnassa, mutta geometrinen radan korjaus toteutettiin myös x – suunnassa. Toteutus tehtiin ilman sumeata säätöä ja muut koeajon parametrin olivat seuraavat:

- Robotin muodon seurannan liikenopeus oli 5 mm/s.
- Anturitiedon luenta suoritettiin 20 kertaa sekunnissa eli keskeytysaliohjelma kutsun (traproutines) arvo oli 0.05.
- Geometrinen korjausarvo asetettiin aluksi siten, että x on z.
- Robotin ohjausjärjestelmän systeemin parametriarvot process update time (0.04834 s) ja queue time (0.193536) säilytettiin aluksi oletusarvoina.

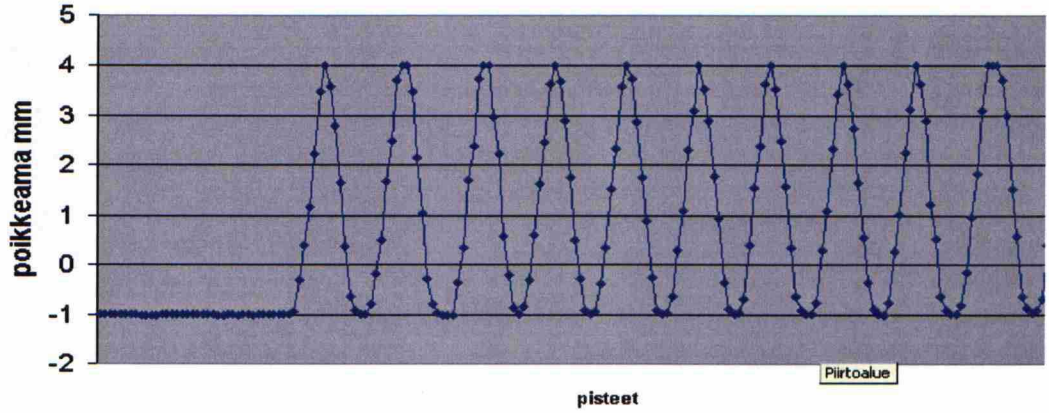


Kuva 6.19 Robotin liikerataa z – suunnassa kuvaavasta käyrästä on nähtävissä, että poikkeama -z-suunnassa kuvaa työkalun painautumista puuhun, mistä johtuu tasainen arvo -1 mm. Voiman ylittäessä annetun arvo, joka oli **200 N**, tapahtuu korjaus z- akselin +suunnassa, mutta kuten käyrästä on nähtävissä, niin korjaus on liian suuri, mitä yritettiin seuraavaksi parantaa.

Testi 2:

Voimaa mitattiin edelleen vain z – suunnassa, mutta geometrista radan korjausta muutettiin x – suunnassa. Toteutus tehtiin ilman sumeata säätöä ja muut koeajon parametrin olivat seuraavat:

- Robotin muodon seurannan liikenopeus oli 5 mm/s.
- Anturitiedon luenta oli edelleen sama eli 20 kertaa sekunnissa.
- Geometrista korjausarvoa muutettiin seuraavasti, $x = 5*z$.
- Robotin systeemin parametriarvot säilytettiin vielä oletusarvoissa.

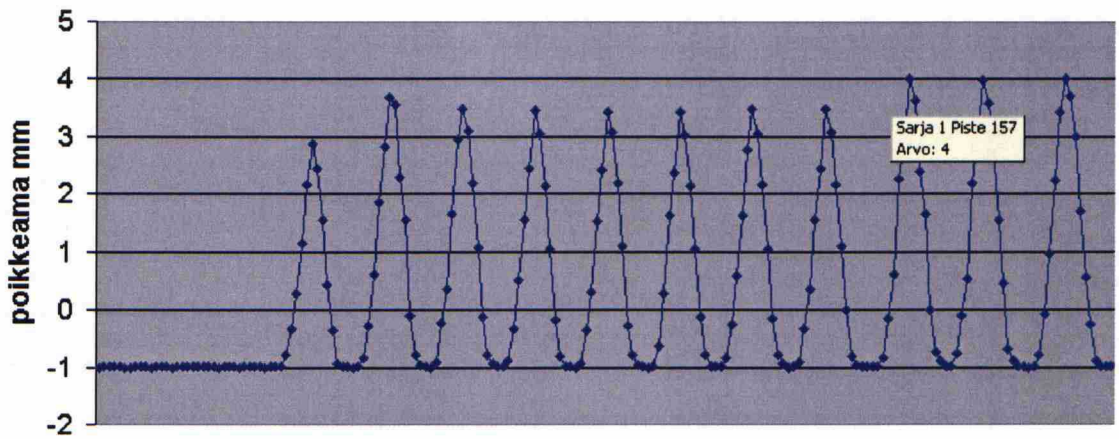


Kuva 6.20 Lisäämällä laskennallinen geometrinen korjaus x – akselin suuntaan eli vaadittu z – liikkeen korjausarvo kerrottiin viidellä. Tällöin liike vähän tasaantui, mutta +suuntainen eli ylöspäin tapahtuva korjaus oli edelleen liian suuri.

Testi 3:

Seuraavaksi muutettiin robottiohjaimen järjestelmän sisäisiä parametriarvoja muiden asioiden säilyessä ennallaan.

- Robotin systeemin parametriarvot olivat process update time **0.02 s**, queue time **0.10**. Process update time määrittelee tavallaan ohjelman päivitysvälin eli laskennan aikavälin ja nyt tämä puolitettiin alkuperäisestä. Queue time määrittelee robotin akselin servomoottorien ohjausarvojen odotusajan, joka myös nyt puolitettiin. Kyseiset muutetut arvot olivat samalla ohjaimen sallimia minimiarvoja.



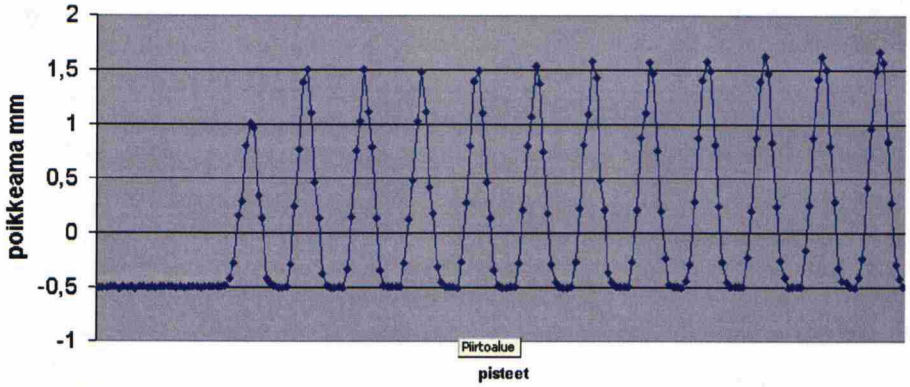
Kuva 6.21 Robotin systeemin parametriarvoja muutettiin CPU:n osalta seuraavaksi siten, että puolitettiin prosessoinnin päivitysaika ja servojen päivitysarvojen jonotusaika samoin. Tällöin poikkeama ylöspäin väheni aavistuksen.

Testi 4:

Kuten edellisestä testistä voidaan päätellä, niin robotin sisäisten parametriarvojen muutokset eivät saaneet aikaiseksi mitään merkittävää parannusta reaaliaikaisessa radan muodostamisessa. Tästä voidaan tehdä johtopäätös, että robotin valmistaja on optimoinut parametriarvot toimituksen yhteydessä jo kohdalleen useisiin robotin käyttökohteisiin. Tässä neljännessä testissä muutettiin geometrinen x-akselin suuntainen korjausarvo vielä kaksinkertaiseksi, ennen kuin tehtiin vastaavat koeajot käyttämällä

sumeata säätöä. Lisäksi pienennettiin aavistuksen kappaleen pintaan vaikuttavan voiman ohjearvoa, joka oli 200 N.

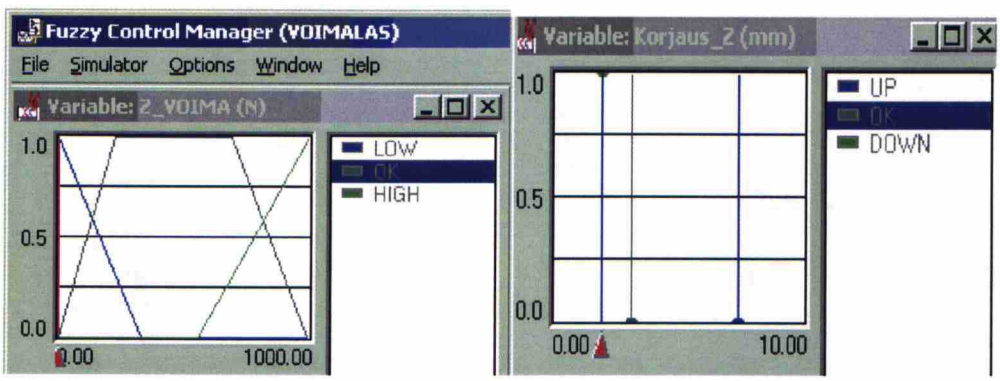
- Robotin muodon seurannan liikenopeus oli 5 mm/s.
- Anturitiedon luenta 20 kertaa sekunnissa eli keskeytysaliohjelma kutsun (traproutines) arvo 0.05.
- Geometrinen korjausarvo oli nyt $x = 10 * z$.
- Robotin systeemin parametriarvot olivat process update time 0.02 s ja queue time 0.10.



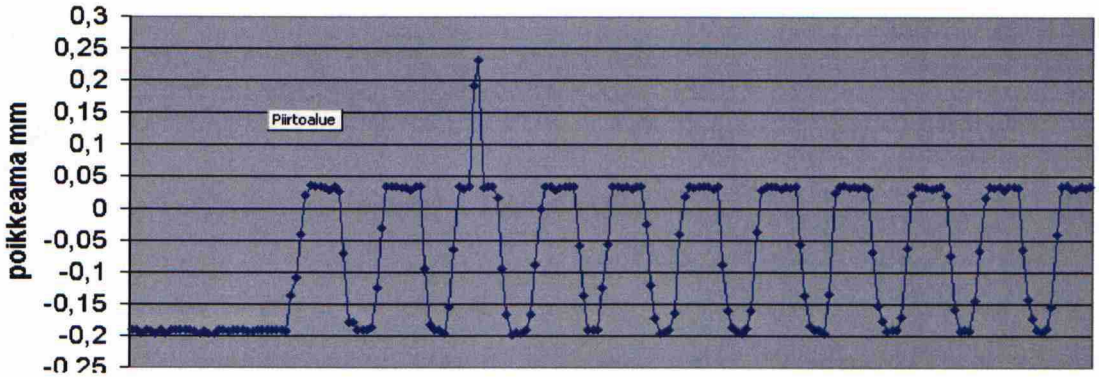
Kuva 6.22 Kappaleen pintaan vaikuttavaa voimaa vähennettiin, jolloin korjaus ylöspäin tapahtui pienemmällä arvolla ja luonnollisesti työkalun painautuminen puuhun pieneni.

Testi 5:

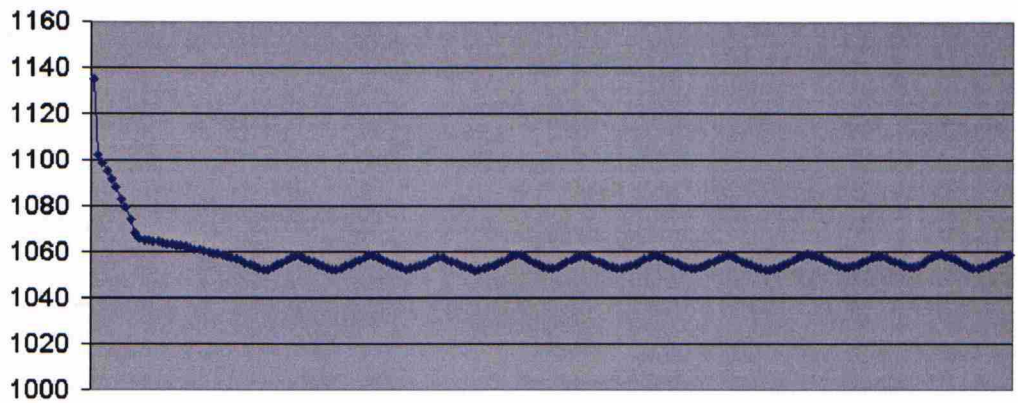
Edelliseen koeajoon lisättiin mukaan nyt sumea säätö. Aluksi vain yhdellä sumealla muuttujalla eli z – suunnassa vaikuttavalla voimalla.



Kuva 6.23 Robotissa olevan voima – anturin viesti on liitetty ohjelmoitavan logiikan AD – korttiin, mutta edellisistä testeistä poiketen se käsiteltiin sumean säädön yksikössä, ennen kuin tarvittava korjausarvo lähetettiin robotin AD – kortille logiikan DA – kortilta. Mitatulle voimalle määriteltiin kolme kielellistä muuttujaa, jotka olivat LOW, OK ja HIGH, joiden perusteella selkeytys toteutettiin. Selkeytyksen lopputulos oli UP, OK tai DOWN eli robotin liikutus z – suunnassa.



Kuva 6.24 Teoreettiset z – suunnan poikkeama-arvot antavat virheellisen kuvan koeajon todellisesta lopputuloksesta, koska taulukosta voisi arvella robotin liikkeen olevan todella tasaista eli vaihtelu -0.2 mm ... 0.05 mm, mutta todellisuudessa jo silmällä oli havaittavissa edelleen huomattavaa heilahtelua ohjatussa z – suunnassa, minkä todentamiseksi robotin ohjelmaan lisättiin z – paikoituspisteiden luenta (kuva 6.25). Kuitenkin jonkin asteista parannusta edellisestä testistä tapahtui, mikä on myös luettavissa kuvan käyrästä eli korjauksen suunnan muutokset tasaantuivat.



Kuva 6.25 Robotin todellinen z – suuntainen liike voidaan nähdä taulukkoon tallennetuista z – koordinaatistoakselin arvoista. Keskimääräisen vaihteluvälin amplitudi oli noin 5 mm.

Yhteenvetoa:

Tässä vaiheessa voidaan todeta se, että sumea säätö tuo jonkin asteisen parannuksen robotin reaaliaikaisen radan muodostamisessa. Sen avulla on helpompi saada aikaiseksi aavistuksen verran tasaisempi robotin liike eli robotin värähtely pienenee vähän. Analysoitaessa näitä suoritettuja koeajoja voitiin tehdä seuraavat huomiot.

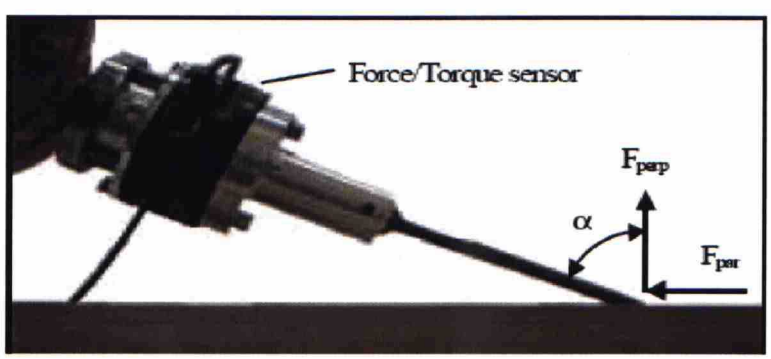
- Ennen kuin radan päivitys tapahtuu ohjaimessa, niin keskeytysaliohjelmalta saadaan kuitenkin 4...6 korjausta eli anturitiedon päivityksen suhteen ei todellisuudessa ole vaikeuksia, vaikka joissakin vertailututkimuksissa näin on kerrottu. Nythän anturitiedot vielä kulkivat logiikan kautta.
- Keskeytysaliohjelman kutsuväli oli siis 0,05 s (1 / 20), joten (4..6) * 0.05 s = 0,2 – 0,3 s, joten radan päivitysväli on noin 5... 3,3 Hz.
- Teoreettinen liike x-suunnassa robotin liikkuessa nopeudella 5 mm/s saadaan seuraavasta laskutoimituksesta: 5 mm/s / 20 Hz = 0,25 mm / keskeytyssuoritus Todellinen liike x – suunnassa radan päivitystä kohden on 5 mm * (0,2...0,3) eli 1...1,5 mm.

Robottiohjelman keskeytysaliohjelma, jossa suoritettiin korjauksen määrittely muodon seurannassa, sisälsi myös anturitietojen tallennuksen taulukkoon ja kun tämä taulukko-

osuus poistettiin kokonaan, niin robotin liikkeen heilahtelu pieneni huomattavasti. Tämä osoittaa jälleen sen tosiasian, että robottiohjaimen laskentakyky on rajallinen, jolloin käytännössä täytyy välttää kaikkea ylimääräistä oheistoimintaa robotin ohjelmassa eli kaikki mahdollinen laskentakapasiteetti ainoastaan robotin liikkeen ohjaukseen. Näin on myös todettu australialaisten [14] tekemissä lihajalostusteollisuuden robottitutkimuksissa.
Tärkeimpänä ja samalla myös tavallaan huonoimpana asiana suoritetut koeajot paljastivat sen tosiasian, että tässä tutkimuksessa käytetyn robotin ohjaimen akselitietokoneet kykenivät päivittämään reaaliaikaisessa radan muodostamisessa uutta rataa ainoastaan 0, 2 s ... 0,3 s väliajoin. Tästä aiheutuu robotin heilahteleva liike, mikä on todettu esimerkiksi jo vuonna 1994 tehdyssä vertailututkimuksessa [17].

6.3 REFERENSSITUTKIMUS [15]

Robotin yhteydessä käytettävälle voimaohjaukselle olisi paljon erilaisia käyttökohteita juuri puusepänteollisuudessa, kuten tämä ruotsalaisten tekemä vertailututkimus osoittaa. Muotoilu on tärkeä tekijä puuteollisuudessa ja ennen kaikkea huonekaluteollisuudessa. Tässä referenssitutkimuksessa on kuvattu menetelmää robotilla tapahtuvaan puun leikkaamiseen. Tällainen mahdollisuus antaisi vapaammat kädet muotoilulle. Ohjausjärjestelmä perustuu voima-anturilta tulevaan takaisinkytkentään. Voima-anturi on liitetty robotin käsivarteen. Kokeilut osoittivat tarvittavan voiman olevan 40 N-100 N riippuen leikkausparanmetreistä ja puulajista.
Puun leikkaus on erikoisen kiinnostava jatkotutkimuksen kohde antureilla ohjatun robottisovelluksen luomiseksi, koska puun leikkaus on eräs parhaiten tutkittuja puuntyöstömenetelmiä. Tässä projektissa yritettiin jatkuvana toimintona mitata leikkausvoimia. Muuttamalla leikkaustaltan kulman asentoa ja suuntaa voidaan syntyviä leikkausvoimia kontrolloida.
Voima-anturi on kiinnitetty robotin käsivarren laippaan, jolloin se välittää välittömästi muutokset momentin suuruudessa puuta leikattaessa. Leikkausvoimat voidaan jakaa kolmeen suuntaan vaikuttaviksi eli 1) kohtisuoraan pinnasta poispäin, 2) vastakkaiseen suuntaan työkalunsuunnasta katsottuna ja 3) voiman normaalin suuntaisiksi.



Kuva 6.26 Robotin työkalulaippaan kiinnitettyyn voima-anturiin vaikuttavat voimat puuta leikattaessa.[15]

Tuloksissa todetaan, että anturilta tulevan viestin taajuus on noin 100 Hz, johon myös robottiohjaimen täytyisi kyetä vastaamaan. Tällöinkin robotin liikenopeudet ja samalla puun leikkauksen nopeudet ovat välillä 5 ... 50 mm/s.

7. JOHTOPÄÄTÖKSET JA TULEVAISUUS

7.1 YLEISTÄ

Teollisuusrobottien käytössä on muistettava tarraimen ja työkalun keskeinen merkitys sovelluksen onnistumisen kannalta. Jos tarraimen suunnittelussa epäonnistutaan, niin tällöin on epäonnistuttu koko robotisointiprojektissa. Robotin tarraimen suunnittelu vaatii monesti erittäin pitkälle menevää monen alan osaamista ja yleensä voidaan sanoa, että vaatimuksena on pitkälle jalostetun mekatronisen laitteen tuntemusta suunniteltaessa robottitarraimia. Sana "mekatroniikka" (mechatronics) voidaan oikeastaan jakaa kolmeen osaan 1) mekaniikka 2) elektroniikka ja 3) tietotekniikka. Antureilla on keskeinen merkitys tarraimien käytössä ja suunnittelussa.
Kehitettäessä uusia käyttösovellutuksia roboteille tarraimet ja siinä käytettävät anturit muodostavat poikkeuksetta ydinkysymykset sovelluksen onnistumisen kannalta. Vaativimmissa kohteissa robotilta vaaditaan suurta adaptiivisuutta eli kykyä mukautua muuttuviin ympäristöolosuhteisiin. Nykyisissä robottiohjaimissa on jo pidemmän aikaan käytössä olleet järjestelmät, joilla robotin sisäinen adaptiivisuus on hoidettu hyvin, mutta käyttäjän halutessa enemmän ulkoista adaptiivisuutta tietyissä sovelluksissa onkin ilmaantunut ratkaisemattomia esteitä. Tähän on syynä ollut robottiohjaimen tietokoneiden rajallinen laskentakyky, mikä on tullut ilmi myös useissa vertailututkimuksissa. Vasta aivan viimeaikaisissa robottiohjaimissa laskentakyky on saatu nostettua tasolle, joka on riittävä reaaliaikaisen radan muodostamisessa. Nyt onkin odotettavissa esimerkiksi robotilta puuttuneen tai ainakin vajavaisen tuntoaistin lisääntyminen robotisoinnin yhteydessä. Robotisoinnin käyttösovelluksia voidaan tällöin lisätä, koska robotti saadaan toimimaan enemmän ihmiskäden mukaisesti saadessaan myös "herkän " tuntoaistin. Tähän ovat kiinnittäneet huomiota myös robottivalmistajat. Kesällä 2005 ABB Oy:n robottitehtaalla Ruotsin Västeråsissa suoritettiin koeajoja voima-anturin avulla toteutettua auton automaattivaihteiston turbiinipyörän asennusta, missä robotti saatiin toimimaan vastaavalla tavalla kuin ihminen on kyseistä työtä suorittanut. Erilaisissa hiontatehtävissä tullaan vastaavanlaisia sovelluksia näkemään. Tällä hetkellä esimerkiksi Suomessa Oras Oy:n vesijohtohanatehtaalla on käyttöönottovaiheessa robotisointiprojekti, jossa voima-anturi onkin sijoitettu robotin yläkäsivarren laippaan, kun se perinteisesti on ollut kiinteässä hiomakoneessa. Vastaavanlaisten sovellusten yleistyessä tulemme tulevaisuudessa näkemään aivan varmasti näitä myös puusepänteollisuuden hiontatehtävissä. Puuteollisuuden piirissä on tunnettu jo pidemmän aikaa suurta mielenkiintoa robotilla suoritettavaan puutuotteiden hiontaa kohtaan.

7.2 TUTKIMUSTYÖN TULOKSET JA JOHTOPÄÄTÖKSET

Tutkimustyössä käytetty ABB:n IRB 4400 -robotti varustettuna ohjaimella S4C on väylä- ja tietoliikenneliitännöiltään monipuolinen ja samoin kuin itse ohjelmointikieleleltään se edustaa myös uudempaa sukupolvea. Tutkimustyön tarkoituksena ollut adaptiivisuuden lisääminen robotin ohjauksen yhteyteen ei kuitenkaan onnistunut aivan tutkimustyön tavoitteissa asetetulla tavalla, koska robotin kyky päivittää uutta

liikerataansa oli liian hidas. Nykyisissä ABB:n robottiohjaimissa itse ohjelmointikieli on pysynyt muuttumattomana, mutta ohjaimen laskentakyky on parantunut huomattavasti, mikä mahdollistaa robotin ulkoisen adaptiivisen ohjauksen kehittämistä aivan eri tasolle kuin nyt suoritetuissa sovelluksissa voitiin tehdä.
Etsittäessä uusia robotisointikohteita puuteollisuudessa oleellinen asia on mielestäni se, että robotti kykenee kunnolliseen reaaliaikaisen radan muodostamiseen anturitietojen perusteella. Robotille asennettava tuntoaisti voima-anturin muodossa mahdollistaa aktiivisen voimaohjauksen. Robotin laippaan asennettavien voima/momenttisovittimien tarjonta on tällä hetkellä hyvin rajallinen. Referenssitutkimuksissa oli käytetty vain kahden eri valmistajan sovittimia. Hannoverin teollisuusmessuilla huhtikuussa 2006 ei löytynyt ensimmäistäkään voima-anturia, jossa olisi yhdistetty useamman suuntainen voiman mittaus kuin yhdessä suunnassa. Nyt robottiohjaimien kehittyessä laskentakyvyltään riittävälle tasolle se antaisi aihetta lähteä kehittämään robotteihin hinnaltaan edullisempia voiman ja momentin mittaukseen tarvittavia työkalulaippoja. Lahdessa sijaitseva Raute Oy on esimerkiksi tunnettu venymäliuska-antureiden ja punnitusantureiden valmistaja, joten osaamista olisi aivan lähietäisyydellä.
Puuteollisuuden robotisoinnin edistämisessä perinteisissä sovelluksissa kuten työstökoneiden ja kuljettimien yhteydessä olisi tietokoneisiin kehitetyillä käyttöliittymillä merkitystä, koska tällöin voitaisiin tarvittavaa robottiohjelmoinnin edellyttämää osaamista madaltaa, jolloin puuteollisuudessa yleensä vaatimattomampi tekninen osaaminen ei muodostuisi kynnyskysymykseksi robottien käyttöön otolle.

Tässä tutkimustyössä robotin ohjauksessa adaptiivisuutta yritettiin kehittää siis laser- ja voima-antureiden avulla, mutta myös lisäämällä ohjauksen yhteyteen nykyaikainen ohjelmoitava logiikka varustettuna sumean säädön yksiköllä. Sumealla säädöllä saatiin aikaiseksi helpommin käyttöön otettavissa oleva ulkoinen adaptiivinen ohjaus-järjestelmä, mutta itse robotin liikkeen lopputulokseen reaaliaikaisessa radan muodostamisessa sillä ei ollut kovinkaan suurta merkitystä. Se ainoastaan aavistuksen verran vähensi robotin värähtelyliikettä.
Tutkimusprojektin aikana hankittu ABB WebWare SDK 4.6 – ohjelmistotyökalu on erinomainen apu kehitettäessä PC – tietokoneisiin käyttöliittymiä. Sen avulla voitiin nyt saada suoraan tietokoneelle antureiden välittämää dataa ja robotin paikoituspisteistä numeroarvoja suoritettaessa koeajoja.

Käytännön valmistelut robotin ja muiden tarvittavien laitteiden osalta aloitettiin jo kesällä 2005 ja niitä kehitettiin koko tutkimustyön suorituksen ajan. Robotilla suoritetuilla koeajoilla saatiin useita pieniä asioita optimoitua ja tässä tutkimustyössä käytännössä suoritetut asiat muodostavatkin työmäärältään ja ajankäytöltään merkittävän osan.

Optimoitaessa robotin suorittamaa liikettä reaaliaikaisessa radan muodostamisessa keskeisessä osassa robotin parametrien osalta olivat sen liikenopeus ja paikoitus-tarkkuus. Paras lopputulos robotin liikkeen tasaisuuden kannalta saavutettiin liikenopeudella, joka oli välillä 25 mm /s ...60 mm/s ja paikoitustarkkuudella (zone – alue) 5 mm. Paikoitustarkkuutta parantamalla esimerkiksi yhteen millimetriin tai huonontamalla 10 mm:iin poikkeamat oikeasta radasta suurenivat.
Luettavan radan korjauksien skaalauskertoimet kuten esimerkiksi *write_offset.z:= (suuntaZ-adaptz)*0.02* vaikuttivat myös oleellisesti. Pienimmät poikkeamat saavutettiin arvolla 0,02. Skaalauskerroin tarkoittaa anturin antaman mittatiedon ja oikean

mittatiedon välisen eron kertomista luvulla, jolloin lopputuloksena saadaan kyseisen robotin akselin ohjaukselle annettavaa korjausta millimetreissä.
Sumean ohjauksen säädöillä oli kuitenkin kaikkein suurimmat vaikutukset optimoidessa muodon seurannan toteuttamista ja etenkin koeajojen alussa suoritetuissa säädöissä.
Kuten jo aikaisemmin on todettu, niin sumea säätö tuo jonkin asteisen parannuksen robotin reaaliaikaisen radan muodostamisessa, mutta lähinnä siten, että sillä on helppo saada aavistuksen tasaisempi robotin liike eli robotin värähtely pienenee.

Analysoitaessa näitä suoritettuja koeajoja voitiin tehdä seuraavat huomiot ja todeta käytännössä laskennallisiin arvoihin perustuen robotin rajallinen kyky päivittää uutta liikerataansa, mikä on erittäin tärkeätä robotin adaptiivisessa ohjauksessa.

- Ennen kuin radan päivitys tapahtuu ohjaimessa, niin keskeytysaliohjelmalta saadaan kuitenkin 4...6 korjausta.
- Keskeytysaliohjelman kutsuväli oli siis 0,05 s (1 / 20), joten (4...6) * 0.05 s = 0,2 – 0,3 s, joten radan päivitysväli on noin 5... 3,3 Hz.
- Teoreettinen liike x-suunnassa robotin liikkuessa nopeudella 5 mm/s saadaan seuraavasta laskutoimituksesta: 5 mm/s / 20 Hz = 0,25 mm / keskeytys
  Todellinen liike x – suunnassa on: 5 mm * (0,2...0,3) eli 1...1,5 mm.

Robottiohjelman keskeytysaliohjelma, jossa suoritettiin korjauksen määrittely muodon seurannassa, sisälsi myös anturitietojen tallennuksen taulukkoon ja kun tämä taulukko-osuus poistettiin kokonaan, niin robotin liikkeen heilahtelu pieneni huomattavasti. Tämä osoittaa jälleen sen tosiasian, että robottiohjaimen laskentakyky on rajallinen, jolloin käytännössä täytyy välttää kaikkea ylimääräistä oheistoimintaa robotin ohjelmassa eli kaikki mahdollinen laskentakapasiteetti ainoastaan robotin liikkeen ohjaukseen.
Tärkeimpänä ja samalla myös tavallaan huonoimpana asiana suoritetut koeajot paljastivat sen tosiasian, että tässä tutkimuksessa käytetyn robotin ohjaimen akselitietokoneet kykenivät päivittämään reaaliaikaisessa radan muodostamisessa uutta rataa ainoastaan 0, 2 s ... 0,3 s väliajoin.
Eräänä mahdollisuutena nähtiin vielä robotin sisäisten parametritietojen optimointi, mutta niiden muutoksilla ei kuitenkaan ollut suurempaa vaikutusta lopputulokseen, mistä voidaan tehdä se johtopäätös, että robotin valmistaja on systeemiparametritiedot kyllä optimoinut useimpiin mahdollisiin käyttösovelluksiin

7.3 TULEVAISUUS

Yleistä
Tämän päivän robottiohjaimet ovat kehittyneet ison harppauksen eteenpäin verrattuna tässä tutkimustyössä käytettyyn ohjaimeen. Tämä koskee myös muiden valmistajien ohjaimia eikä pelkästään ABB:n kehittämää uuden sukupolven IRC5 – robottiohjainta.
Tämän hetkisillä robottiohjaimilla pystytään jo käyttökelpoiseen reaaliaikaiseen radan muodostamiseen. Aikaisemman sukupolven robottiohjaimien käyttö reaaliaikaisessa radan muodostamisessa on rajoittunut ainoastaan tutkimustöihin ja tulevaisuuden

visioiden luomiseen uusissa robotisointikohteissa. Robottiohjaimien kehitykselle voidaan olettaa edelleen voimakasta jatkoa, koska noin kymmenen vuoden kestoinen lähes paikallaan pysynyt ohjaimien laskentakyky on nyt jäänyt taakse.

Seuraavaksi esitellään lyhyesti kahden robottivalmistajan uuden teknologian robottiohjaimia. Ensin esitellään tärkeimpiä ominaisuuksia ABB:n IRC5 ohjaimesta.

Kuva 7.1 IRC5 – ohjain on myös ulkoisesti jaettu kahteen osaan, joista ylempi on päätason robottiohjain ja alempi on robotin akselien ohjain eli liikkeen ohjaukseen tarkoitettu.

Robottimaailmassa ABB on jo hyvin tunnettu ja tunnustettu siitä, että se on eräs johtavista robottiohjaimen kehittäjistä juuri liikkeen ohjausteknologiassa. Yhdellä uudella IRC5 – ohjaimella voidaan samanaikaisesti ohjata neljää robottia, mikä oleellisesti helpottaa usean robotin käyttöä samassa solussa, kun robottien liikkeitä synkronoidaan keskenään.

Toisena mainittakoon Mitsubishi Electric Oy:n robottiohjain CR2A, joka on tarkoitettu 3 kg ...10 kg SCARA- ja nivelvarsiroboteille. Sen sydämenä toimii 64 – bittinen RISC mikroprosessori. Ohjain kykenee suoriutumaan samanaikaisesti 32 eri ohjelmasta. Ohjauksen adaptiivisuuden kannalta tärkeänä ominaisuutena mainittakoon Ethernetin kautta välitettävän anturitiedon reaaliaikaisuus, jolloin nyt laskentakykyinen tietokoneyksikkö pystyisi todella reaaliaikaiseen radan muodostamiseen.

Mahdolliset uudet tutkimustyöt

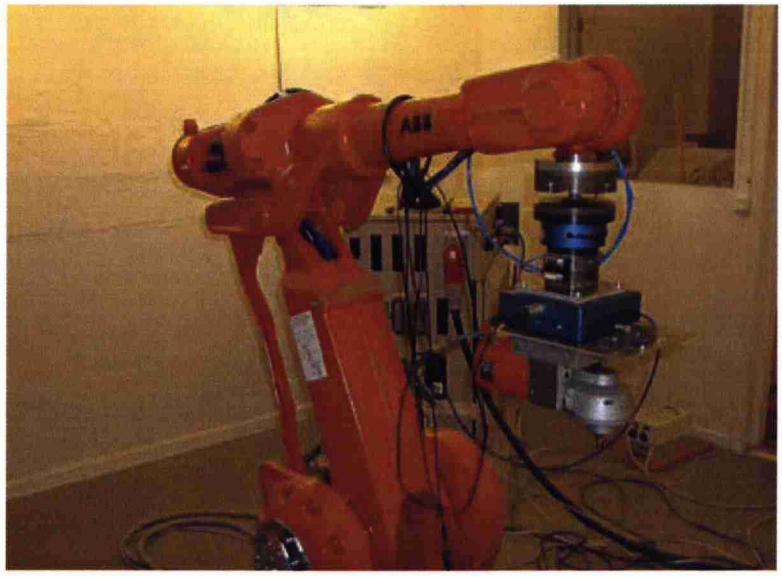
Tämän tutkimustyön ensisijainen tarkoitus oli robotin adaptiivisuuden kehittäminen juuri voimaohjauksen avulla eli lisätä robottiin ulkoinen tuntoaisti. Robottiin tarkoitettu voimaohjauksen sovitin oli nyt omaa valmistetta, jonka suunnittelu ja valmistus vei ehkä liian paljon aikaa. Toisaalta se jäi myös vähän vajavaiseksi momentin mittauksen puuttuessa, koska käytössä oli vain kolme venymäliuska-anturia, jolloin voitiin mitata vain x-, y- ja z-akselien suuntaiset voimat, eikä vastaavien kiertoliikkeiden momentteja ollenkaan. Tosin alussa suoritetuissa alkukoeajoissa tapahtui vielä vahinko, jossa yksi anturi vielä menetettiin, mutta se ei loppujen lopuksi haitannutkaan niin paljon kuin

aluksi oletettiin. Testikoeajotyökalua jouduttiin kuitenkin vielä kehittämään, jottei toista vahinkoa enää pääsisi tapahtumaan, missä myös onnistuttiin.
Jos sama tutkimustyö aloitettaisiin uudestaan, niin lähtökohtana olisi kyllä valmiin voima-anturisovittimen hankinta. Toisaalta niiden erittäin suppea tarjonta eli muutama valmistaja maailmassa herätti ajatuksen uudenlaisesta yhteistoimintaprojektista. Mielestäni nyt olisi tosi otollinen aika lähteä kehittämään robotin yhteyteen tarkoitettuja voiman ja momentin mittaukseen sopivia työkalulaippoja, joissa olisi vähintään kuuden suunnan mittaus. Tällainen yhteistoimintaprojekti olisi mahdollista luoda Lahden ammattikorkeakoulun Tekniikan laitoksen ja Lahdessa sijaitsevan Raute Precision Oy:n kesken. Raute Precision Oy on tunnettu ja merkittävä punnitusantureiden valmistaja ja omaa suuren kokemuksen venymäliuska-antureiden käytöstä ja Tekniikan laitoksen robottilaboratorio olisi oivallinen testausympäristö tämän tutkimustyön myötä lisääntyneen osaamisen vuoksi. Tällöin voisi myös selvittää mahdollisen uudemman robottiohjaimen soveltuvuutta samalle manipulaattorilaitteelle, koska oletettavasti se ei ole oleellisesti muuttunut, eikä ainakaan ulkoisesti eroa laisinkaan vastaavasta tänä päivänä myytävänä olevasta ABB IRB 4400 – robotista. Tämän tapaiset projektit vaativat kylläkin useamman alueen erikoisosaajia, jotka sitoutuvat tosissaan saamaan myös tuloksia aikaiseksi ja tavoitteena täytyy olla siis täysin valmis käyttöön otettavissa oleva robottiin tarkoitettu "tuntoaisti". Tämän mahdollisen projektitoteutuksen jälkeen voitaisiin seuraavaksi lähteä tutkimaan yksityiskohtaisemmin puutuotteiden hiontaa perustuen robotin adaptiivisuuden hyödyntämiseen.
Tosin aivan viimeaikaisissa yhteistyöprojekteissa espanjalaisten ja ruotsalaisten yliopistojen kesken on toteutettu robottitutkimusprojekteja, joissa robotin adaptiivisuutta on kehitetty yhteistoiminnallisella voima- ja kiihtyvyysanturisovitinta käyttäen. Seuraavaksi esitellään keskeisimmät asiat tässä vuonna 2005 tehdyssä tutkimuksessa, jonka nimi on "Voima- ja kiihtyvyysanturin yhteensovittaminen mukautuvassa robotin liikkeen ohjauksessa".

Referenssitutkimus [30]

Tässä vertailututkimuksessa on toteutettu sovellus, jossa on käytetty dynaamisen voimaohjauksen periaatetta, jossa on käytetty useamman ulkoisen anturin yhteistoimintaa lisäämään robotin adaptiivisuutta. Toisena anturina on käytetty tässä tutkimustyössä jo aiemmin esiteltyä JR3–voima-anturisovitinta ja toinen on kapasitiivinen kiihtyvyysanturi. Kiihtyvyysanturin käytöllä on haluttu erottaa robotin työkalulaippaan kohdistuvat hitausvoimat niistä voimista, jotka aiheutuvat työkalulaippaan sen koskettaessa ulkoista pintaa kuten esimerkiksi hiottavaa kappaletta.
Tavallaan tässä vertailututkimuksessa on yhdistetty vielä kolmaskin anturityyppi, joka on jokaisessa robotin akselissa mittaamassa nivelakselien välistä kulmaa eli valosähköistä pulssianturia (encoder). Tämän on mahdollistanut se, että kyseisellä tutkimusryhmällä on ollut käytössään avoimella ohjauksella varustettu ABB:n uuden sukupolven robottiohjain. Näiden kolmen anturin avulla toteutetulla adaptiivisella ohjauksella on tarkoituksena saavuttaa kolmessa karteesisen koordinaatiston suunnassa juuri oikean suuruinen ja tarkka kosketusvoima käsiteltävään kohteeseen.
Tämä useamman anturin käyttö on ollut jo kauemman aikaa tunnettuna vaikeana ongelmana soveltaa sitä robotiikassa. Tutkimustyön tekijöiden mukaan ainoastaan voima-antureiden käyttöön perustuvalla ohjauksella on ongelmana se, että dynaamisissa tilanteissa merkittävässä asemassa ei ole pelkästään robotin kosketuksesta kohteeseen aiheutuneet voimat, vaan myös robotin liikkeestä aiheutuneet hitausvoimat.

Useamman erityyppisen anturin antamien mittaviestien yhdistämistä siten, että niiden viestien käsittelyn jälkeen saadaan lopputuloksena yksi viesti tai ohjaustieto, voitaisiin tavallaan kutsua älykkääksi anturiksi.



Kuva 7.2 Tutkimustyön robottina käytettiin ABB IRB 2400 teollisuusrobottia varustettuna avoimella ohjauksella. Voima-anturisovitin on sijoitettu robotin ranteen laipan ja mukautuvan työkalulaipan välille. Työkalulaippaan on sijoitettu hiomakone ja kiihtyvyysanturi, joka kykenee mittaamaan kiihtyvyydet kaikissa kolmessa maailman koordinaatiston suunnissa. Maksimi kiihtyvyysarvo kyseiselle anturille on 20 g:tä.

Ruotsalaisessa Lundin yliopiston robottilaboratoriossa suoritetuissa käytännön tutkimuksissa oleellista oli myös se, että heillä oli käytössään poikkeuksellisesti avoin robottiohjain, jolloin he pystyivät käyttämään ohjaimen kehitystyössä Matlab/Simulink-ohjelmistoja. Mainittakoon tässä yhteydessä se, että australialainen professori Peter Corke on julkaissut vapaasti saatavilla olevan robotiikan laajennusmoduulin Matlab-ohjelmistoon. Tätä laajennusmoduulia itse vähän käyttäneenä tämän oman tutkimustyön alkumetreillä havaitsin sen käyttökelpoisuuden kylläkin rajoittuneen vain perinteiseen yliopistomaailman PUMA -robotin yhteyteen. [29]

Tämän tutkimustyön tarkoitushan oli kehittää kolmen erillisen anturityypin yhdistelmä, jolla voidaan robotin adaptiivista ohjausta edelleen kehittää vaativiin ja monimutkaisiin robotisointikohteisiin. Samoin he kehittivät robottiohjainta tätä uutta älykästä anturia tukevaksi. Tutkimustyön tuloksia arvioidessaan he katsoivat onnistuneensa luomaan robotin ranteen laippaan uuden tyyppisen ja käyttökelpoisen anturisovittimen, jolla voidaan saada aikaiseksi juuri oikean voiman suuruinen kosketus kohteeseen, koska sillä pystytään myös erottamaan robotin liikkeestä aiheutuvat hitausvoimat. Lisäksi yhdistettäessä sen toimintaan robotin nivelakselien sen hetkiset arvot voidaan robotin mukautuvaa ohjausta edelleen parantaa.

Lähdeluettelo:

[1] Kuivanen, Risto. Robotiikka. Vantaa, 1999. 188 s. ISBN 951-9438-58-0.

[2] Puolakka, Heikki. Sumea logiikka käytännön sovelluksissa. Helsinki, 1997. 285 s. ISBN 951-719-823-X.

[3] Airila, Mauri. Mekatroniikka. Helsinki, 1994. 397 s. ISBN 951-672-173-7.

[4] Niku, Saeed Benjamin. Introduction to Robotics. Analysis, Systems, Application. New Jersey, 2001. 387 s. ISBN 0-13-061309-6.

[5] Klafter, Richard, Chmielewski, Thomas ja Negin, Michael. Robotic Engineering. An Integrated Approach. USA, 1989. ISBN 0-13-782053-4.

[6] Haage, Mathias. RAPID Force Extension Proposal. Tutkimusraportti. Department of Computer Systems. Lund Institute of Technology, 2003.
Saatavilla: http://www.robot.lth.se/proj/autofett/fspec.pdf

[7] Olsson, Tomas. Feedback Control and Sensor Fusion of Vision and Force. Tutkimusraportti. Department of Automatic Control Lund Institute of Technology, 2004.
Saatavilla: http://www.control.lth.se/database/publications/article.pike?artkey=tol04lic

[8] Lundqvist, Rasmus ja Söreling, Tobias. New Interface for Rapid Feedback Control on ABB-Robots. Undergraduate thesis, Linköping University, Department of Mechanical Engineering. Linköping, 2003.
Saatavilla: http://www.diva-portal.org/diva/getDocument?urn_nbn_se_liu_diva-2762-1__fulltext.pdf

[9] Andersson, Jan-Erik ja Johansson, Gert. Adaptive robot control in wood manufacturing industries. The Department of Mechanical Engineering, Linköpings Universitet. Linköping Sweden, 2001.
Saatavilla: http://www.ps.ikp.liu.se/staff/janan/Publications/ICPR-1999.pdf

[10] Haage, Mathias. Extending an industrial robot controller with a fast open sensor interface—implementation and applications. Licentiate Thesis, Department of Computer Science Lund Institute of Technology Lund University. Lund, 2004. ISSN 1652-469.

[11] Magdy, M. Enhancement of sliding mode controller by fuzzy logic with application to robotic manipulators. Faculty of Engineering, Design and Production Engineering Department, Ain Shams University. Kairo, 2004.

[12] Clarence, W. de Silva. Applications of fuzzy logic in the control of robotic manipulators Industrial Automation Laboratory, Department of Mechanical Engineering, The University of British Columbia, Vancouver, 1994.
Saatavilla: http://portal.acm.org/citation.cfm?id=203068.203077

[13] Llu, M.-H. Force- controlled fuzzy-logic-based robotic deburring. Department of Control Engineering, Control System Theory and Robotics Group, Technical University of Darmstadt. Darmstadt, 1994.
Saatavilla: http://www.ingentaconnect.com/content/els/09670661/1995/00000003/00000002/art00076

[14] Li, Zeng, Ring, Peter, MacRae, Kym ja Hinsch, Andrew .Control of Industrial Robots for Meat Processing Applications. Food Science Australia, CSIRO.

[15] Andersson, Jan-Erik ja Johansson Gert. Creation of a Generic Sensor Model for Wood Carving with Industrial Robots.Department of Mechanical Engineering Linköpings Universitet Sweden.

[16] Pires, Norbert J., Ramming, John, Rauch Stephen ja Araûjo, Ricardo. Force/Torque Sensing Applied to Industrial Robotic Deburring. Submitted to Sensor Review Journal, MCB University Press, UK.
Saatavilla: http://robotics.dem.uc.pt/norberto/microsoft2002/docs/ft_sensor.pdf

[17] Kai-Tai, Song ja Hsi-Pin, Li. A Fuzzy Adaptive Control Design for Compliant Motion of a Manipulator. Department of Control Engineering, National Chiao Tung University and Mechanical Industry Research Laboratories, Industrial Technology Research Institute. Taiwan, Republic of China, 1994 IEEE.

[18] ABB Automation Technology Products AB Robotics. Product Specification RobotWare Options for BaseWare OS 3.1.Västerås, 1998.

[19] ABB Automation Technology Products AB Robotics. Product Manual S4C. Västerås, 1998.

[20] ABB Automation Technology Products AB Robotics. Product Specification IRB 4400. Västerås, 1998.

[22] ABB Automation Technology Products AB Robotics. Ethernet Services User's Guide. Västerås, 1998.

[23] ABB Automation Technology Technologies AB Robotics. Installation and Setup Guide WebWare SDK Version 4.6. Document ID: 3HAC 025460-001, SE-721 68. Västerås, 2004.

[24] ABB Automation Technology Technologies AB Robotics. WebWare SDK Controls Reference. USA, 2004.

[25] OMRON SYSMAC CS Series, Programmable Controllers, Programming Manual, Cat. No. W394-E1-07 Revised July 2004.

[26] OMRON, SYSMAC CS Series, Programmable Controllers, Operation Manual, Cat. No. W339-E1-09 Revised July 2004.

[27] OMRON, C200H-FZ001 Fuzzy Logic Unit, Operation Manual, Cat. No. W208-E1-2 Revised September 1999

[28] Fonselius, Jaakko, Pekkola, Kari, Selosmaa, Seppo, Ström, Markku ja Välimaa, Taisto. Automaatiolaitteet. Helsinki, 1999. 199 s. ISBN 951-37-1834-4.

[29] Corke, P.I. A Robotics Toolbox for MatLab. IEEE Robotics and Automation Magazine, Volume 3, 1996. Saatavilla: http://www.cat.csiro.au/ict/staff/pic/robot/

[30] Gamez Garcia, J., Gomez Ortega, J., Robertsson, A. and Johansson, R. Force and Acceleration Sensor Fusion for Compliant Robot Motion Control. System Engineering and Automation Dept. Jaen University, Spain and Dept. of Automatic Control Lund University, Sweden.
International Conference on Robotics and Automation Barcelona, Spain, April 2005, IEEE 2005.

LIITTEET

LIITE 1 *Robotin ohjelma laserantureilla suoritetuissa koeajoissa*

Ensin esitellään ohjelmassa käytetyt muuttujat, vakiot ja persistantit. Persistant tyyppisiä muuttujia joudutaan käyttämään silloin, kun datatietoa luetaan tai kirjoitetaan robottiohjelman toisesta moduulista tai kuten tässä sovelluksessa datatietoa on käytetty Visual Basic käyttöliittymältä.

Datat (muuttujat, vakiot ja persistantit):

Robotin paikoituspisteet:

```
CONST robtarget lopetusp2:=*
CONST robtarget orginalap
VAR robtarget aloitusZ: =*
CONST robtarget lopputesti: =*
CONST robtarget lahtopaikka: =*
VAR robtarget lopetusp1:=*
 VAR robtarget aloitusp1:=*
```

Num tyyppiset datat:

```
PERS num kplpintaZ: =*
PERS num kplpintaY: =*
PERS pos posaloitusz: =*
PERS num tcptarkkuus: =5;
PERS num tcpnopeus: =50;
PERS num suuntaZ: =0.1648;
PERS num suuntaY: =-0,0213623;
VAR num i;
VAR num j;
PERS num adaptz: =53;
PERS num newadaptz: =53;
PERS num adapty: =57;
PERS num newadapty: =57;
VAR num taulu_Z{300};
VAR num taulu_Y{300};
```

Muut:

```
 VAR pos write_offset;
 VAR pos total_offset;
 VAR intnum intno1;
 VAR corrdescr corrdescr_Z;
 VAR corrdescr corrdescr_Y;
 VAR corrdescr corrdescr_X;
 VAR zonedata tarkkuus: =[FALSE, 5,8,8, 0.8,8,0.8];
 VAR speeddata nopeus: =[60, 500, 5000, 1000];
 PERS bool robostart: =TRUE;
 PERS num suorituslaskuri: =1;
VAR iodev zarvot;
VAR iodev yarvot;
```

```
VAR iodev numfile;
VAR iodev numfile2;
CONST string arvotz: ="valuesz.xls";
CONST string arvoty: ="valuesy.xls";
CONST string numarvoz:="numz.txt";
CONST string numarvoy: ="numy.txt";
```

Keskeytysaliohjelmaa kutsutaan 20 kertaa sekunnissa reaaliaikaisen radan muodostamiseksi. Anturitiedot luetaan ja paikoituksen korjaus suoritetaan jokaisella keskeytysohjelman suorituskerralla.

TRAP ReadSensors

```
suuntaY: = AInput(Y_SUUNTA);        "Luetaan" analogiatulo Y_laserilta
write_offset.z:=0;
write_offset.x:=0;
write_offset.y:=(suuntaY - adapty)*0.02; "Lasketaan" tarvittava Y-suuntainen
                                         korjaus
CorrWrite corrdescr_Y, write_offset;    Toteutetaan Y-suuntainen korjaus
 taulu_Y{j}:=write_offset.y;            Sijoitetaan korjattuarvo taulukkoon
 j:=j+1;
suuntaZ: = AInput(Z_SUUNTA);           "Luetaan" analogiatulo Z_laserilta
write_offset.y:=0;
write_offset.x:=0;
write_offset.z:=(suuntaZ-adaptz)*0.1;   "Lasketaan" tarvittava Z-suuntainen
                                        korjaus
CorrWrite corrdescr_Y, write_offset;    Toteutetaan Z-suuntainen korjaus
taulu_Z{i}:=write_offset.z;
i:=i+1;
write_offset.z:=0;
write_offset.y:=0;
write_offset.x:=tcpnopeus/20;
CorrWrite corrdescr_X,write_offset;
IDelete intno1;                         Lopetetaan keskeytyksen suoritus
CONNECT intno1 WITH ReadSensors;        Kytketään keskeytys
                                        keskeytysaliohjelmaan
ITimer\Single, 0.05,intno1;             Kutsutaan kerran 0.05 s kuluttua
                                        keskeytystä
ENDTRAP
```

PROC main ()

```
                                        Ohjelmamoduulin pääohjelma
adapty:=47;                             Adaptiivisuuden "oikeat" lähtöarvot
 adaptz:=49;
 reg2:=2;
 reg4:=2;
suorituslaskuri:=0;
uusi:
suuntaZ:=0;
suuntaY:=0;
kplpintaY:=76;
kplpintaZ:=61;
 IF robostart THEN
SetDO DO16JAKORdx8,1;
ELSE
SetDO DO16JAKORdx8,0;
```

```
 ENDIF
 WaitUntil robostart=TRUE;
SetDO DO16JAKORdx8,1;
MoveL lahtopaikka, v100,z10,testlaser;
IF suorituslaskuri>=1 GOTO kierto;
MoveL orginalap, v200,z50,testlaser;
aloitusZ:=orginalap;
MoveL aloitusZ, v20, fine, testlaser;
WHILE kplpintaZ>=70 DO              Etsitään kappaleen pinta ensin Z-
                                    suunnassa ja luetaan paikoitusarvot
MoveL Offs(aloitusZ, 0,0,-2), v5,fine, testlaser;
aloitusZ:=CRobT(\Tool:=testlaser\WObj:=wobj0);
ENDWHILE
WHILE kplpintaY<70 DO               Seuraavaksi suoritetaan sama Y -
MoveL Offs(aloitusZ, 0,-2,0), v5,fine, testlaser;      suunnassa
aloitusZ:=CRobT(\Tool:=testlaser\WObj:=wobj0);
 ENDWHILE
WHILE kplpintaY>75 DO
 MoveL Offs (aloitusZ, 0, 2, 0), v5, fine, testlaser;
 aloitusZ: =CRobT (\Tool:=testlaser\WObj:=wobj0);
ENDWHILE
aloitusp1:=aloitusZ;                Tallennetaan radan aloituspaikka
lopetusp1:=Offs(aloitusp1,550,1,5);  "Lasketaan" radan lopetuspaikka
WaitTime 1;
kierto:
 posaloitusz:=CPos(\Tool:=testlaser\WObj:=wobj0);
 i:=1;
 j:=1;
 nopeus.v_tcp:=tcpnopeus;           Annetaan liikenopeus ja paikoitusten
 tarkkuus.pzone_tcp:=tcptarkkuus;   tarkkuus
suuntaZ:=AInput(Z_SUUNTA);
suuntaY:=AInput(Y_SUUNTA);
IF kplpintaY<47 OR kplpintaZ<30 Stop;
CorrCon corrdescr_Z;
CorrCon corrdescr_Y;
CorrCon corrdescr_X;
CONNECT intno1 WITH ReadSensors;  Kutsutaan ensimmäisen kerran
 ITimer\Single, 0.05,intno1;       keskeytysaliohjelma 0.05 s jälkeen
 MoveL aloitusp1,v50,z10,testlaser;  Aloitetaan adaptiivinen radan
MoveL lopetusp1,nopeus, tarkkuus, testlaser\Corr;  muodostaminen
 total_offset: =CorrRead();
TPWrite "korjaus y_suunnassa"\Num:=total_offset.y;
TPWrite "korjaus z_suunnassa"\Num:=total_offset.z;
WaitTime 3;
CorrDiscon corrdescr_Z;
CorrDiscon corrdescr_Y;
CorrDiscon corrdescr_X;
CorrClear;
 IDelete intno1;
WaitTime 0.5;
MoveL Offs(lopetusp1,0,0,200),v100,z10,testlaser;
MoveJ lahtopaikka, v100,z10,testlaser;
WaitTime 3;
suuntaZ:=0;
suuntaY:=0;
Open tietokone\File:=arvotz, zarvot;        Avataan tietokoneyhteys, jotta
FOR i FROM 1 TO 300 DO                      voidaan siirtää korjausarvojen
Write zarvot, " "\Num:=taulu_Z{i};          Taulukko MS EXCEL - taulukko-
ENDFOR                                      laskentaohjelmaan
```

```
WaitTime 1;
Close zarvot;
FOR j FROM 1 TO 300 DO
Open tietokone\File:=arvoty, yarvot;
Write yarvot," "\Num:=taulu_Y{j};
ENDFOR
Close yarvot;
Open tietokone\File:=numarvoz, numfile\Read; Avataan tietokoneyhteys
uudelleen,
reg1:=0.1*ReadNum(numfile);          jotta voidaan lukea MS Excelin
 IF reg1<reg2 THEN                   suorittamat laskennat paremmiksi
newadaptz:=49-reg1;                  adaptiivisuusarvoiksi
ELSE
newadaptz:=49+reg1;
ENDIF
 adaptz: =newadaptz;
reg2:=reg1;
Close numfile;
Open tietokone\File:=numarvoy,numfile2\Read;
 reg3:=0.1*ReadNum(numfile2);
IF reg3<reg4 THEN
 newadapty: =47-reg3;
 ELSE
 newadapty: =47+reg3;
 ENDIF
adapty: =newadapty;
reg4:=reg3;
Close numfile2;
SetDO DO16JAKORdx8,0;
robostart: =FALSE;
suorituslaskuri:=suorituslaskuri+1;
 GOTO uusi;
Stop;
ENDPROC
ENDMODULE
```

LIITE 2 *Robotin ohjelma voima-antureilla suoritetuissa koeajoissa*

Tässä liitteessä on esitetty käytetty ohjelma täydellisenä paikoituspisteineen. Muuten ohjelman rakenne noudattelee liitteessä 1 esiteltyä ohjelmaa.

MODULE FORCE

```
CONST robtarget ylos: = [[62.96,-1387.32, 1135.44],[0.051535,-0.130538,-0.983366,0.115313],[-1,0,1,0]];
CONST robtarget loppupiste: = [[61.41,-1386.94, 1065.58],[0.040246,-0.03595,-0.991256,0.120423],[-1,0,1,0];
CONST robtarget aloituspiste: = [[-213.4,-1386.32, 1058.03],[0.034763,0.009782,-0.991843,0.122251],[-2,0,1,0]];
CONST robtarget lahestymispiste: = [-214.51,-1386.79, 1076.16],[0.0386,-0.02264,-0.991664,0.120829],[-2,0,1,0]];
CONST robtarget kotipiste: = [[-245.18,-934.45, 1198.65],[0.114632,-0.72043,-0.681774,0.055015],[-2,-1,2,0]];
VAR pos write_offset;
VAR pos total_offset;
VAR intnum intno1:=0;
VAR num z;
VAR num x;
VAR num y;
VAR num taulu_Z {1000};
VAR num taulu_X {1000};
VAR num i;
VAR num j;
VAR corrdescr corrdescr_total;
VAR iodev zarvot;
VAR iodev yarvot;
CONST string arvotz:="valuesz.xls";
CONST string arvoty:="valuesy.xls";
CONST string numarvoz:="numz.txt";
CONST string numarvoy:="numy.txt";
```

TRAP ReadSensors

```
 y:=0;
 z:=(Z_SUUNTA-10)*0.05;
 x:=10*z;
 taulu_X{j}:=x;
 j:=j+1;
 taulu_Z{i}:=z;
 i:=i+1;
 write_offset:=[x, y, z];
 CorrWrite corrdescr_total, write_offset;
 IDelete intno1;
 CONNECT intno1 WITH ReadSensors;
 ITimer\Single,0.05,intno1;                    Täytyy olla paras mahdollinen eli 0.05
ENDTRAP
```

PROC PathRoutine ()

```
CorrCon corrdescr_total;
CONNECT intno1 WITH ReadSensors;
ITimer\Single, 0.05, intno1;
MoveL Offs (lahestymispiste, 0,0,-10), v100,z10,testi_voima;
MoveL aloituspiste, v5,z1,testi_voima;
MoveL loppupiste, v5, fine, testi_voima\Corr;
MoveL ylos, v10, fine, testi_voima;
total_offset:= CorrRead ();
TPWrite "korjaus on "\Pos:=total_offset;
WaitTime 5;
CorrDiscon corrdescr_total;
CorrClear;
IDelete intno1;
RETURN;
```

```
  ENDPROC

PROC main ()
  i:=1;
  j:=1;
  MoveJ kotipiste, v200,z50,testi_voima;
  MoveL lahestymispiste, v200,z50,testi_voima;
  PathRoutine;
  Open tietokone\File:= arvotz, zarvot;
  FOR i FROM 1 TO 300 DO
    Write zarvot," "\Num:=taulu_Z{i};
  ENDFOR
  WaitTime 1;
  Close zarvot;
  FOR j FROM 1 TO 300 DO
  Open tietokone\File:= arvoty, yarvot;
  Write yarvot," "\Num:=taulu_X{j};
  ENDFOR
  Close yarvot;
  MoveL Offs (ylos,0,0,10),v50,z50,testi_voima;
  MoveL lahestymispiste, v200,z50,testi_voima;
  Stop;
 ENDPROC
ENDMODULE
```

LIITE 3 *Ohjelmoitavan logiikan ohjelma voima-antureilla suoritetuissa koeajoissa*

Ohjelmoitavan logiikan ohjelma on luotu Omron CX - One – ohjelmalla. Logiikkaohjelma käsittelee ensin voima-antureilta tulevan jänniteviestin AD – muuntimella. Ohjelmalla myös muutetaan eli skaalataan AD – muuntimen tuottamat heksadesimaaliluvut Visual Basic – käyttöliittymälle oikeiksi desimaalilukuarvoiksi. Ohjelmoitavan logiikan ohjelma sisältää myös sumealle säädölle tarvittavat ohjaukset.

0 0	' ALUSSA ANALOGIALÄHTÖJEN LÄHTÖJEN NOLLAUS LD P_First_Cycle MOV(021) #0 2010
1 2	LD P_First_Cycle MOV(021) #0 2011
2 4	LD P_First_Cycle MOV(021) #0 2012
3 6	LD P_First_Cycle MOV(021) #0 2013
4 8	LD P_On MOV(021) D705 2013
5 10	LD P_On SCL(194) D704 D300 D2011
6 12	LD P_On MOV(021) #0 D300
7 14	LD P_On MOV(021) #7D7 D301
8 16	LD P_On MOV(021) #10 D302
9 18	LD P_On MOV(021) #FA0 D303
10 20	LD P_On BIN(023) D2011 2011
11 22	' TÄSTÄ ALKA Z- SUUNNAN OHJAUS VOIMA-ANTURILLA LD P_First_Cycle MOV(021) #11 D12
12 24	LD P_On *B(424) 2003 #5 D105
13 26	LD P_On /B(434) D105 D12 D5

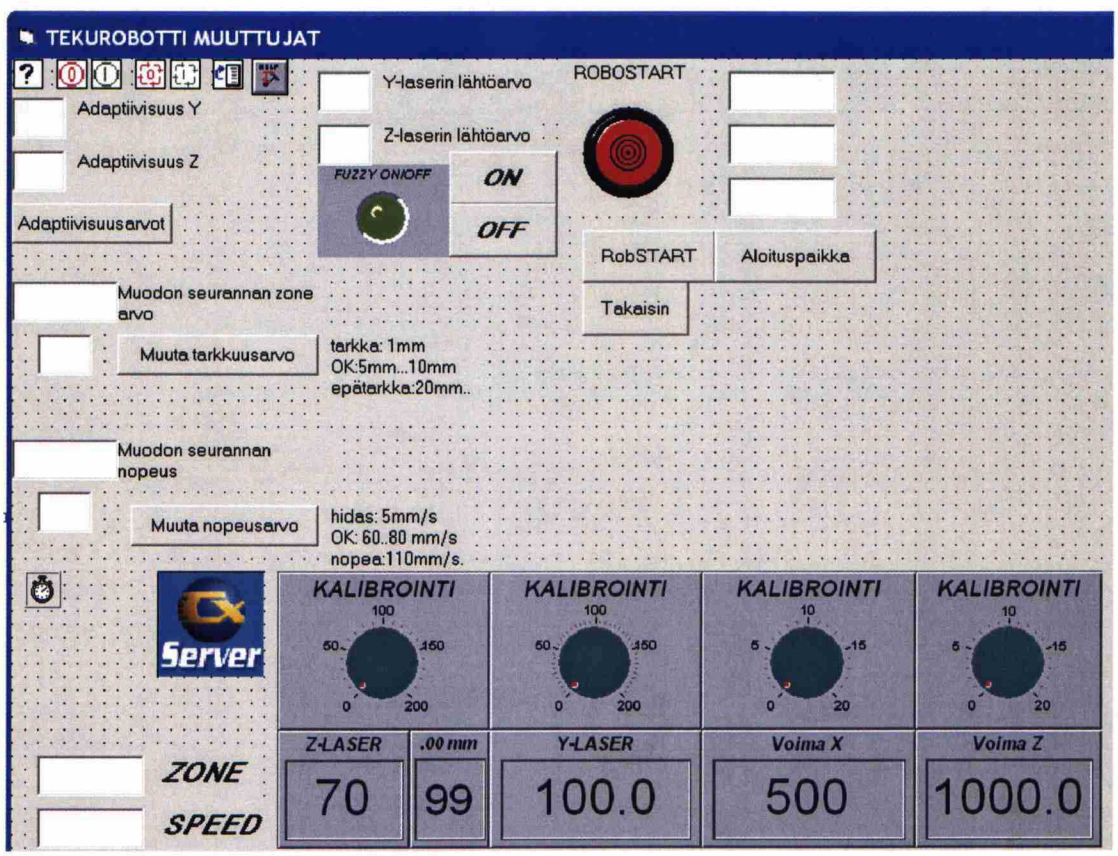
```
14      ' POIS KÄYTÖSTÄ SUMEAN YHTEUDESSÄ
   28   LD P_Off
        // Siirretään Voima-anturin Z arvo eli CIO 2003 dmkanavaan 203
        *(420) D5 #2 2012
15      LD 2020.15
   30   *B(424) D5 #5 D233
16      LD 2020.15
   32   BIN(023) D233 D203
17      ' TÄSTÄ ALKA X- SUUNNAN OHJAUS VOIMA-ANTURILLA
   34   LD P_First_Cycle
        MOV(021) #9 D10
18      LD P_On
   36   *B(424) 2001 #5 D101
19      LD P_On
   38   /B(434) D101 D10 D1
20      ' POIS KÄYTÖSTÄ SUMEAN YHTEYDESSÄ
   40   LD P_Off
        // Siirretään Voima-anturin x arvo eli CIO 2001 dmkanavaan 201
        *(420) D1 #2 2010
21      LD 2020.15
   42   *B(424) D1 #30 D211
22      LD 2020.15
   44   BIN(023) D211 D201
23      ' LASEREIDEN KÄYTTÖ ALKAA TÄSTÄ, Z-laserin mitta-arvon muutos Visual Basic:a varten
   46   LD P_On
        /B(434) D2004 #100 D2044
24      ' POIKKEAMA MITATTUNA Z-LASERILLA 0.1 s välein
   48   LD P_0_1s
        MOV(021) D2004 D4002
25      LD P_On
   50   CMP(020) D2004 D4002
```

```
26
     52  LD P_GT
         -B(414) D2004 D4002 D4000

27
     54  LD P_LT
         -B(414) D4002 D2004 D4000

28
     56  ' POIKKEAMA KORJAUKSEN ALKU ELI KUN OLLAAN URASSA
         LD P_GT
         LD P_EQ
         KEEP(011) 42.04

29
     59  LD P_On
         // Siirretään LASER  Z arvo eli CIO 2004 eli dm 704 dm-kanavaan 204
         MOV(021) D704 D204

30
     61  LD P_On
         BIN(023) 2004 D704

31
     63  LD P_On
         SCL(194) D704 D24 D2004

32
     65  LD P_On
         MOV(021) #4000 D24

33
     67  LD P_On
         MOV(021) #FFF D25

34
     69  LD P_First_Cycle
         MOV(021) #6000 D26

35
     71  LD P_First_Cycle
         MOV(021) #0 D27

36
     73  ' Y-laserin mitta-arvon muutos Visual Basic:a varten
         LD P_On
         /B(434) D2005 #1 D2055

37
     75  LD P_On
         // Siirretään LASER  Y arvo eli CIO 2005 eli dm 705 dm-kanavaan 205
         MOV(021) D705 D205

38
     77  LD P_On
         BIN(023) 2005 D705
```

```
39      LD P_On
   79   SCL(194) D705 D34 D2005

40      LD P_On
   81   MOV(021) #300 D34

41      LD P_On
   83   MOV(021) #FA0 D35

42      LD P_First_Cycle
   85   MOV(021) #450 D36

43      LD P_First_Cycle
   87   MOV(021) #7C0 D37

44      ' ROBOTIN OHJAUS JA SUMEAN SÄÄDÖN KÄYTTÖ
   89   LD P_On
        MOV(021) #7 2020
        // Kanavaan 2020(20n0) ilmoitetaan analogiatulojen määrä: eli 7kpl, 1. on voimaY,  2. VoimaX, 3.VoimaZ, 4.LaserZ, 5. LaserY,

45      LD P_On
   91   MOV(021) #201 2021
        // Kanavaan 2021 ilmoitetaan  sen sanan osoite, joka ensimmäisen analogiatulon datan. 0-->DM-alue ja ekakanava 201.

46      ' Kanavaan 2022 ilmoitetaan analogialähtöjen määrä eli nyt 3 kpl
   93   LD P_On
        MOV(021) #3 2022

47      LD P_On
   95   MOV(021) #110 2023
        // Kanavaan 2023 ilmoitetaan  sen sanan osoite, joka on  ensimmäisen analogialähtödata. 0-->DM-alue ja D110 ensimmäinen

48      ' SUMEAN SÄÄDÖN KÄYNNISTUS TAPAHTUU Visual Basic lomakkeelta
   97   LD 30.01
        LDNOT 30.01
        KEEP(011) 30.00
        // Aloitetaan tai lopetetaan sumeaohjaus

49      LD P_On
  100   OUT 2020.15

50      ' Sumeiden lähtöjen siirto analogialähtöihin
  102   LD P_On
        MOV(021) D112 2012

51      LD P_On
  104   MOV(021) D110 2010
```

LIITE 4 *Visual Basic - käyttöliittymän lomake ja sen tarvitsemat koodit*

Robotin adaptiivisuutta ja reaaliaikaista radan muodostamista tutkittaessa suoritettiin erilaisia testikoeajoja, joissa käytettiin laserantureita tai voima-antureita. Näitä varten luotiin Visual Basic-ohjelmalla käyttöliittymä, joka oleellisesti helpotti näiden koeajojen suoritusta ja tiedon keruuta.
Se mahdollisti myös muuttujatasolla tapahtuvan kommunikoinnin robotin ja ohjelmoitavan logiikan ohjelmien välille. Alla olevassa kuvassa on nähtävissä käyttöliittymän visuaalinen ilme ja käytetyt objektit, jotka on OPC- rajapinnan välityksellä liitetty joko robotin ohjelmaan tai ohjelmoitavan logiikan ohjelmaan.



Käyttöliittymälomakkeen koodit:

```
Private Y_Laser As Single
Private Z_Laser As Single
Private speed As Single
Private zone As Single

Private Sub cmdAdapt_Click ()          Voidaan tarvittaessa nähdä ns. adaptiivisuusarvot
  Dim adaptiveY As Single
  Dim adaptiveZ As Single
  Dim intRetVal1 As Integer
  Dim intRetVal2 As Integer
  intRetVal1 = Helper1.S4ProgramNumVarRead ("adaptz", 0, adaptiveY)
  intRetVal2 = Helper1.S4ProgramNumVarRead ("adapty", 0, adaptiveZ)
  txtAdaptY.Text = adaptiveY
  txtAdaptZ.Text = adaptiveZ
```

```
End Sub
Private Sub cmdBack_Click ()                    Lopetetaan ja palataan aloituslomakkeelle
  Muuttujat.Hide
  Form1.Show
End Sub


Private Sub cmdPos_Click ()                     Painamalla painiketta aloitusarvot saadaan näkyviin
                                                contour tracking -toiminnon alkupiste (x, y ja z)
  Dim status As Integer
  Dim robpos As S4Pos
  Set robpos = CreateObject ("S4Pos")
  status = Helper1.S4ProgramVariableRead ("posaloitusz", 0, robpos)
  If status = 0 Then
     txtPOSX.Text = Str (robpos.Pos (0))
     txtPosY.Text = Str (robpos.Pos (1))
     txtPOSZ.Text = Str (robpos.Pos (2))
    Else
    MsgBox "status=" + Str (status)
  End If
End Sub


Private Sub cmdRoboStart_Click ()               Aloitetaan testikoeajon suoritus
  Dim uusibool As Boolean
  Dim intRetVal3 As Integer
  Dim ResultSpec As Integer
  Dim ResultID As Long
  'Dim intRetVal2 As Integer
  uusibool = True
  intRetVal3 = Helper1.S4ProgramBoolVarWrite ("robostart", 0, uusibool, ResultSpec, ResultID)
  'intRetVal2 = Helper1.S4ProgramBoolVarRead ("robostart", 0, uusibool)
End Sub

Private Sub cmdStartFUZZY_Click ()              Käynnistetään sumean säädön toiminto
  Comms1.CIO ("30.01") = True
End Sub

Private Sub cmdStopFuzzy_Click ()               Pysäytetään sumean säädön toiminto
  Comms1.CIO ("30.01") = False
End Sub

Private Sub cmdZone_Click ()                    Muutetaan robotin paikoituspisteiden tarkkuusarvoa
  Dim uusiarvo As Single
  Dim intRetVal1 As Integer
  Dim ResultSpec As Integer
  Dim ResultID As Long
  'Dim zone As Single
  Dim intRetVal2 As Integer
  uusiarvo = Val(txtChangeZone.Text)
  intRetVal1 = Helper1.S4ProgramNumVarWrite ("tcptarkkuus", 0, uusiarvo, ResultSpec, ResultID)
  intRetVal2 = Helper1.S4ProgramNumVarRead ("tcptarkkuus", 0, zone)
  txtZone.Text = zone
End Sub


Private Sub Form_Load ()                        Kun lomake avautuu, niin luetaan käyttöliittymässä
                                                oleviin textbox – kenttiin sen hetkiset arvot
  Dim lasersuuntaY As Single
  Dim lasersuuntaZ As Single
```

```
    Dim intRetVal1 As Integer
    Dim intRetVal2 As Integer
    Dim zone As Single
    Dim intRetVal3 As Integer
    Dim speed As Single
    Dim intRetVal4 As Integer
    Dim adaptiveY As Single
    Dim adaptiveZ As Single
    Dim intRetVal5 As Integer
    Dim intRetVal6 As Integer
    intRetVal5 = Helper1.S4ProgramNumVarRead ("adaptz", 0, adaptiveY)
    intRetVal6 = Helper1.S4ProgramNumVarRead ("adapty", 0, adaptiveZ)
    txtAdaptY.Text = adaptiveY
    txtAdaptZ.Text = adaptiveZ
    intRetVal1 = Helper1.S4ProgramNumVarRead ("suuntaY", 0, lasersuuntaY)
    intRetVal2 = Helper1.S4ProgramNumVarRead ("suuntaZ", 0, lasersuuntaZ)
    txtLaserY.Text = lasersuuntaY
    txtLaserZ.Text = lasersuuntaZ
    intRetVal3 = Helper1.S4ProgramNumVarRead ("tcptarkkuus", 0, zone)
    txtZone.Text = zone
    intRetVal4 = Helper1.S4ProgramNumVarRead ("tcpnopeus", 0, speed)
    txtSpeed.Text = speed
End Sub


Private Sub tcpSpeed_Click ()          Muutetaan robotin liikkeen nopeusarvoa
    Dim uusiarvo As Single
    Dim intRetVal1 As Integer
    Dim ResultSpec As Integer
    Dim ResultID As Long
    'Dim speed As Single
    Dim intRetVal2 As Integer
    uusiarvo = Val(txtChangeSpeed.Text)
    intRetVal1 = Helper1.S4ProgramNumVarWrite "tcpnopeus", 0, uusiarvo, ResultSpec, ResultID)

    intRetVal2 = Helper1.S4ProgramNumVarRead ("tcpnopeus", 0, speed)
    txtSpeed.Text = speed
End Sub


Private Sub Timer1_Timer ()          Arvojen päivitysvälin toteutus
    Dim lasersuuntaY As Single
    Dim lasersuuntaZ As Single
    Dim intRetVal1 As Integer
    Dim intRetVal2 As Integer
    intRetVal1 = Helper1.S4ProgramNumVarRead ("suuntaY", 0, lasersuuntaY)
    intRetVal2 = Helper1.S4ProgramNumVarRead ("suuntaZ", 0, lasersuuntaZ)
    txtLaserY.Text = lasersuuntaY
    txtLaserZ.Text = lasersuuntaZ

    Comms1.D ("207") = Val (txtSpeed.Text)
    Comms1.D ("206") = Val (txtZone.Text)
    txtFuzzy5.Text = Comms1.D ("207")
    txtFuzzy4.Text = Comms1.D ("206")
End Sub
```

```
' Luetaan kappaleen pinnat eli Y- ja Z -laserien arvot logiikan D2055 ja D2044 kanavista ja
' siirretään ne kellopulssin avulla robotille num tyyppisille muuttujille arvoY ja arvoZ,
' jolloin voidaan suorittaa kalibrointi ja robotin ns. OK asennon löytyminen ja aloittaa
' contour tracking toiminta robotilla

Private Sub Timer2_Timer ()
   Dim arvoY As Single
   Dim arvoZ As Single
   Dim robarvoY As Single
   Dim robarvoZ As Single
   Dim intRetVal1 As Integer
   Dim intRetVal2 As Integer
   Dim ResultSpec As Integer
   Dim ResultID As Long

   Y_Laser = CInt (Comms1.D ("2055"))
   Text1.Text = CInt (((Y_Laser / 16) & (CInt (ModY_Laser * 10) / 16))) * 10)
   arvoY = CInt (((Y_Laser / 16) & (CInt ((ModY_Laser * 10) / 16))) * 10)
   intRetVal1 = Helper1.S4ProgramNumVarWrite ("kplpintaY", 0, arvoY, ResultSpec, ResultID)

   Z_Laser = CInt (Comms1.D ("2044"))
   Text2. Text = Val (CInt (((Z_Laser / 16) & (CInt ((ModZ_Laser * 10) / 16))) * 10))
   arvoZ = CInt (((Z_Laser / 16) & (CInt ((ModZ_Laser * 10) / 16))) * 10)
   intRetVal2 = Helper1.S4ProgramNumVarWrite ("kplpintaZ", 0, arvoZ, ResultSpec, ResultID)

   intRetVal3 = Helper1.S4ProgramNumVarRead ("kplpintaY", 0, robarvoY)
   intRetVal4 = Helper1.S4ProgramNumVarRead ("kplpintaZ", 0, robarvoZ)
   txtrobarvoY.Text = robarvoY
   txtrobarvoZ.Text = robarvoZ
End Sub
```